Minimum Bisection is Fixed Parameter Tractable

Marek Cygan * Daniel Lokshtanov † Marcin Pilipczuk ‡ Michał Pilipczuk §
Saket Saurabh¶

Abstract

In the classic Minimum Bisection problem we are given as input a graph G and an integer k. The task is to determine whether there is a partition of $V(G)$ into two parts A and B such that $||A| - |B|| \leq 1$ and there are at most k edges with one endpoint in A and the other in B. In this paper we give an algorithm for Minimum Bisection with running time $\mathcal{O}(2^{\mathcal{O}(k^3)}n^3 \log^3 n)$. This is the first fixed parameter tractable algorithm for Minimum Bisection. At the core of our algorithm lies a new decomposition theorem that states that every graph G can be decomposed by small separators into parts where each part is "highly connected" in the following sense: any cut of bounded size can separate only a limited number of vertices from each part of the decomposition.

Our techniques generalize to the weighted setting, where we seek for a bisection of minimum weight among solutions that contain at most k edges.

*Institute of Informatics, University of Warsaw, Poland, cygan@mimuw.edu.pl.

†Department of Informatics, University of Bergen, Norway, daniello@ii.uib.no.

‡Department of Informatics, University of Bergen, Norway, Marcin.Pilipczuk@ii.uib.no.

§Department of Informatics, University of Bergen, Norway, michal.pilipczuk@ii.uib.no.

¶Institute of Mathematical Sciences, India saket@imsc.res.in Department of Informatics, University of Bergen, Norway, Saket.Saurabh@ii.uib.no.

1 Introduction

In the MINIMUM BISECTION problem the input is a graph G on n vertices together with an integer k, and the objective is to find a partition of the vertex set into two parts A and B such that $|A| = \lfloor \frac{n}{2} \rfloor$, $|B| = \lceil \frac{n}{2} \rceil$, and there are at most k edges with one endpoint in A and the other endpoint in B. The problem can be seen as a variant of MINIMUM CUT, and is one of the classic NP-complete problems [14]. MINIMUM BISECTION has been studied extensively from the perspective of approximation algorithms [12, 11, 19, 25], heuristics [4, 6] and average case complexity [3].

In this paper we consider the complexity of MINIMUM BISECTION when the solution size k is small relative to the input size n. A naïve brute-force algorithm solves the problem in time $n^{\mathcal{O}(k)}$. Until this work, it was unknown whether there exists a *fixed parameter tractable* algorithm, that is an algorithm with running time $f(k)n^{\mathcal{O}(1)}$, for the MINIMUM BISECTION problem. In fact MINIMUM BISECTION was one of very few remaining classic NP-hard graph problems whose parameterized complexity status was unresolved. Our main result is the first fixed parameter tractable algorithm for MINIMUM BISECTION.

Theorem 1.1. MINIMUM BISECTION *admits an* $\mathcal{O}(2^{\mathcal{O}(k^3)}n^3 \log^3 n)$ *time algorithm.*

Theorem 1.1 implies that MINIMUM BISECTION can be solved in polynomial time for $k = \mathcal{O}(\sqrt[3]{\log n})$. In fact, our techniques can be generalized to solve the more general problem where the target $|A|$ is given as input, the edges have non-negative weights, and the objective is to find, among all partitions of $V(G)$ into A and B such that A has the prescribed size and there are at most k edges between A and B, such a partition where the total weight of the edges between A and B is minimized.

Our methods. The crucial technical component of our result is a new graph decomposition theorem. Roughly speaking, the theorem states that for any k, every graph G may be decomposed in a tree-like fashion by separators of size $2^{\mathcal{O}(k)}$ such that each part of the decomposition is "highly connected". To properly define what we mean by "highly connected" we need a few definitions. A *separation* of a graph G is a pair $A, B \subseteq V(G)$ such that $A \cup B = V(G)$ and there are no edges between $A \setminus B$ and $B \setminus A$. The *order* of the separation (A, B) is $|A \cap B|$. A vertex set $X \subseteq V(G)$ is called (q, k)-*unbreakable* if every separation (A, B) of order at most k satisfies $|(A \setminus B) \cap X| \leq q$ or $|(B \setminus A) \cap X| \leq q$. The parts of our decomposition will be "highly connected" in the sense that they are $(2^{\mathcal{O}(k)}, k)$-unbreakable. We can now state the decomposition theorem as follows.

Theorem 1.2. *There is an algorithm that given* G *and* k *runs in time* $\mathcal{O}(2^{\mathcal{O}(k^2)}n^2m)$ *and outputs a tree-decomposition* (T, β) *of* G *such that* (i) *for each* $a \in V(T)$, $\beta(a)$ *is* $(2^{\mathcal{O}(k)}, k)$*-unbreakable in* G, (ii) *for each* $ab \in E(T)$ *we have that* $|\beta(a) \cap \beta(b)| \leq 2^{\mathcal{O}(k)}$, *and* $\beta(a) \cap \beta(b)$ *is* $(2k, k)$*-unbreakable in* G.

Here $\beta(a)$ denotes the bag at node $a \in V(T)$; the completely formal definition of tree-decompositions may be found in the preliminaries. It is not immediately obvious that a set X which is (q, k)-unbreakable is "highly connected". To get some intuition it is helpful to observe that if a set X of size at least $3q$ is (q, k)-unbreakable then removing any k vertices from G leaves almost all of X, except for at most q vertices, in the same connected component. In other words, one cannot separate two large chunks of X with a small separator. From this perspective Theorem 1.2 can be seen as an approximate way to "decompose a graph by k vertex-cuts into it's $k+1$-connected components" [7], which is considered an important quest in structural graph theory. The proof strategy of Theorem 1.2 is inspired by the recent decomposition theorem of Marx and Grohe [16] for graphs excluding a topological subgraph. Contrary to the approach of Marx and Grohe [16], however, the crucial technical tool we use to decompose the graph are the important separators of Marx [20].

Our algorithm for Minimum Bisection applies Theorem 1.2 and then proceeds by performing bottom up dynamic programming on the tree-decomposition. The states in the dynamic program are similar to the states in the dynamic programming algorithm for Minimum Bisection on graphs of bounded treewidth [17]. Property (ii) of Theorem 1.2 ensures that the size of the dynamic programming table is upper bounded by $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$. For graphs of bounded treewidth all bags have small size, making it easy to compute the dynamic programming table at a node b of the decomposition tree, if the tables for the children of b have already been computed. In our setting we do not have any control over the size of the bags, we only know that they are $(2^{\mathcal{O}(k)}, k)$-unbreakable. We show that the sole assumption that the bag at b is $(2^{\mathcal{O}(k)}, k)$-unbreakable is already sufficient to efficiently compute the table at b from the tables of its children, despite no a priori guarantee on the bag's size. The essence of this step is an application of the "randomized contractions" technique [9].

We remark here that the last property of the decomposition of Theorem 1.2 — the one that asserts that adhesions $\beta(a) \cap \beta(b)$ are $(2k, k)$-unbreakable in G — is not essential to establish the fixed-parameter tractability of Minimum Bisection. This high unbreakability of adhesions is used to further limit the number of states of the dynamic programming, decreasing the dependency on k in the algorithm of Theorem 1.1 from double- to single-exponential.

Related work on balanced separations. There are several interesting results concerning the parameterized complexity of finding balanced separators in graphs. Marx [20] showed that the the vertex-deletion variant of the bisection problem is W[1]-hard. In Minimum Vertex Bisection the task is to partition the vertex set into three parts A, S and B such that $|S| \leq k$ and $|A| = |B|$, and there are no edges between A and B. It is worth mentioning that the hardness result of Marx [20] applies to the more general problem where $|A|$ is given as input, however the hardness of Minimum Vertex Bisection easily follows from the results presented in [20].

As the vertex-deletion variant of the bisection problem is W[1]-hard, we should not expect that our approach would work also in this case. Observe that one can compute the decomposition of Theorem 1.2 and define the states of the dynamic programming over the tree decomposition, as it is done for graphs of bounded treewidth. However, we are unable to perform the computations needed for one bag of the decomposition. Moreover, it is not only the artifact of the "randomized contractions" technique, but the hard instances obtained from the reduction of [20] are in fact highly unbreakable by our definition, and Theorem 1.2 would return a trivial decomposition.

Feige and Mahdian [13] studied cut problems that may be considered as approximation variants of Minimum Bisection and Minimum Vertex Bisection. We say that a vertex (edge) set S is an α-(edge)-separator if every connected component of $G \setminus S$ has at most αn vertices. The main result of Feige and Mahdian [13] is a randomized algorithm that given an integer k, $\frac{2}{3} \leq \alpha < 1$ and $\epsilon > 0$ together with a graph G which has an α-separator of size at most k, outputs in time $2^{f(\epsilon)k}n^{\mathcal{O}(1)}$ either an α-separator of size at most k or an $(\alpha + \epsilon)$-separator of size strictly less than k. They also give a deterministic algorithm with similar running time for the edge variant of this problem. To complement this result they show that, at least for the vertex variant, the exponential running time dependence on $1/\epsilon$ is unavoidable. Specifically, they prove that for any $\alpha > \frac{1}{2}$ finding an α-separator of size k is W[1]-hard, and therefore unlikely to admit an algorithm with running time $f(k)n^{\mathcal{O}(1)}$, for any function f. On the other hand, our methods imply a $2^{\mathcal{O}(k^3)}n^{\mathcal{O}(1/\alpha)}$ time algorithm for finding an α-edge-separator of size at most k, for any $\alpha > 0$.

Minimum Bisection on planar graphs was shown to be fixed parameter tractable by Bui and Peck [5]. It is interesting to note that Minimum Bisection is not known to be NP-hard on planar graphs, and the complexity of Minimum Bisection on planar graphs remains a challenging open problem. More recently, van Bevern et al. [27] used the treewidth reduction technique of Marx et al. [21] to give a fixed parameter tractable algorithm for Minimum Bisection for the special case

when removing the cut edges leaves a constant number of connected components. Their algorithm also works for the vertex-deletion variant the same restrictions. Since Minimum Vertex Bisection is known to be W[1]-hard, it looks difficult to extend their methods to give a fixed parameter tractable algorithm for Minimum Bisection without any restrictions. Thus, Theorem 1.1 resolves an open problem of van Bevern et al. [27] on the existence of such an algorithm.

Related work on graph decompositions. The starting point of our decomposition theorem is the "recursive understanding" technique pioneered by Grohe et al. [15], and later used by Kawarabayashi and Thorup [18] and by Chitnis et al. [9] to design a number of interesting parameterized algorithms for cut problems. Recursive understanding can be seen as a reduction from a parameterized problem on general graphs to the same problem on graphs with a particular structure. Grohe et al. [15] essentially use recursive understanding to reduce the problem of deciding whether G contains H as a topological subgraph to the case where G either excludes a clique on $f(|H|)$ vertices as a *minor* or contains at most $f(|H|)$ vertices of degree more than $f(|H|)$, for some function f. Marx and Grohe [16] subsequently showed that any graph which excludes H as a topological subgraph can be decomposed by small separators, in a tree-like fashion, into parts such that each part either excludes a clique on $f(|H|)$ vertices as a minor or contains at most $f(|H|)$ vertices of degree more than $f(|H|)$, for some function f. Thus, the decomposition theorem of Marx and Grohe [16] can be seen as a "structural" analogue of the recursive understanding technique for topological subgraph containment.

Both Kawarabayashi and Thorup [18] and Chitnis et al. [9] apply recursive understanding to reduce certain parameterized cut problems on general graphs to essentially the same problem on a graph G where $V(G)$ is $(f(k), k)$-unbreakable for some function f. Then they proceed to show that the considered problem becomes fixed parameter tractable on $(f(k), k)$-unbreakable graphs. Observe that Minimum Bisection on $(f(k), k)$-unbreakable graphs is trivially fixed parameter tractable - if the number of vertices is more than $2f(k)$ we can immediately say no, while if the number of vertices is at most $2f(k)$, then a brute force algorithm is already fixed parameter tractable. More importantly, it turns out that even the more general problem where $|A|$ is given on the input can be solved in fixed parameter tractable time on $(f(k), k)$-unbreakable graphs via an application of the "randomized contractions" technique of Chitnis et al [9]. It is therefore very natural to try to use recursive understanding in order to reduce Minimum Bisection on general graphs to Minimum Bisection on $(f(k), k)$-unbreakable graphs.

Unfortunately, it seems very difficult to pursue this route. In particular, recursive understanding works by cutting the graph into two parts by a small separator, "understanding" the easier of the two parts recursively, and then replacing the"understood" part by a constant size gadget. For Minimum Bisection it seems unlikely that the understood part can be emulated by any constant size gadget because of the balance constraint in the problem definition. Intuitively, we would need to encode the behaviour of the understood part for every possible cardinality of A, which gives us amount of information that is not bounded by a function of k. The issue has strong connections to the fact that the best known algorithm for Minimum Bisection on graphs of bounded treewidth is at least quadratic [17] rather than linear.

At this point our decomposition theorem comes to the rescue. It precisely allows us to *structurally* decompose the graph in a tree-like fashion into $(f(k), k)$-unbreakable parts, which provides much more robust foundations for further algorithmic applications. Thus, essentially our decomposition theorem does the same for cut problems as the decomposition theorem of Marx and Grohe [16] does for topological subgraph containment. Notably, the "recursive understanding" step used by Kawarabayashi and Thorup [18] and Chitnis et al. [9] for their problems could be replaced by dynamic programming over the tree-decomposition given by Theorem 1.2.

We remark here that it has been essentially known, and observed earlier by Chitnis, Cygan

and Hajiaghayi (private communication), that Minimum Bisection can be solved in FPT time on sufficiently unbreakable graphs via the "randomized contractions" technique. Furthermore, although our application of this framework to handle one bag of the decomposition is more technical than in [9], due to the presence of the information for children bags, it uses no novel tools compared to [9]. Hence, we emphasize that our main technical contribution is the decomposition theorem (Theorem 1.2), with the fixed-parameter algorithm for Minimum Bisection being its corollary via an involved application of known techniques.

Organisation of the paper. After setting up notation and recalling useful results on (important) separators in Section 2, we turn our attention to the decomposition theorem and prove Theorem 1.2 in Section 3. The algorithm for Minimum Bisection in the unweighted setting, promised by Theorem 1.1, is presented in Section 4. We discuss the weighted extension in Section 5 and how to find an α-edge-separator of size at most k in Section 6. Section 8 concludes the paper.

The first 10 pages of the paper contain, apart from the introduction and preliminaries, an almost complete proof of Theorem 1.2, the main technical contribution of this paper. Some proofs of claims (marked with ♠) that are either simple or only auxiliary (e.g., needed to improve the running time) are postponed to Section 7. The proof of Theorem 1.1 is a non-trivial application of the "randomized contractions" approach of Chitnis et al [9] on top of the decomposition given by Theorem 1.2. Due to its highly technical character, this proof resides later in the paper.

2 Preliminaries

We use standard graph notation, see e.g. [10]. We use n and m to denote cardinalities of the vertex and edge sets, respectively, of a given graph provided it is clear from the context. We begin with some definitions and known results on separators and separations in graphs.

Definition 2.1 (separator). For two sets $X, Y \subseteq V(G)$ a set $W \subseteq V(G)$ is called an $X - Y$ separator if in $G \setminus W$ no connected component contains a vertex of X and a vertex of Y at the same time.

Definition 2.2 (separation). A pair (A, B) where $A \cup B = V(G)$ is called a *separation* if $E(A \setminus B, B \setminus A) = \emptyset$. The *order* of a separation (A, B) is defined as $|A \cap B|$.

Definition 2.3 (important separator). An inclusion-wise minimal $X - Y$ separator W is called an *important* $X - Y$ separator if there is no $X - Y$ separator W' with $|W'| \leq |W|$ and $R_{G \setminus W}(X \setminus W) \subsetneq R_{G \setminus W'}(X \setminus W')$, where $R_H(A)$ is the set of vertices reachable from A in the graph H.

Lemma 2.4 ([8, 22])**.** *For any two sets $S, T \subseteq V(G)$ there are at most 4^k important $S-T$ separators of size at most k and one can list all of them in $\mathcal{O}(4^k k(n+m))$ time.*

We proceed to define tree-decompositions. For a rooted tree T and a non-root node $t \in V(T)$, by parent(t) we denote the parent of t in the tree T. For two nodes $u, t \in T$, we say that u is a *descendant* of t, denoted $u \preceq t$, if t lies on the unique path connecting u to the root. Note that every node is thus its own descendant.

Definition 2.5 (tree decomposition). A *tree decomposition* of a graph G is a pair (T, β), where T is a rooted tree and $\beta : V(T) \to 2^{V(G)}$ is a mapping such that:

- for each node $v \in V(G)$ the set $\{t \in V(G) | v \in \beta(t)\}$ induces a nonempty and connected subtree of T,
- for each edge $e \in E(G)$ there exists $t \in V(T)$ such that $e \subseteq \beta(t)$.

The set $\beta(t)$ is called the *bag at* t, while sets $\beta(u) \cap \beta(v)$ for $uv \in E(T)$ are called *adhesions*. Following the notation from [16], for a tree decomposition (T, β) of a graph G we define auxiliary mappings $\sigma, \gamma : V(T) \to 2^{V(G)}$ as

$$\gamma(t) = \bigcup_{u \preceq t} \beta(u), \qquad \sigma(t) = \begin{cases} \emptyset & \text{if t is the root of } T \\ \beta(t) \cap \beta(\text{parent}(t)) & \text{otherwise} \end{cases}$$

Finally, we proceed to the definition of unbreakability.

Definition 2.6 (**(q,k)-unbreakable set).** We say that a set A is (q,k)-*unbreakable*, if for any separation (X, Y) of order at most k we have $|(X \setminus Y) \cap A| \leq q$ or $|(Y \setminus X) \cap A| \leq q$. Otherwise A is (q,k)-*breakable*, and any separation (X, Y) certifying this is called a *witnessing separation*.

Let us repeat the intuition on unbreakable sets from the introduction. If a set X of size at least $3q$ is (q,k)-unbreakable then removing any k vertices from G leaves almost all of X, except for at most q vertices, in the same connected component. In other words, one cannot separate two large chunks of X with a small separator.

Observe that if a set A is (q,k)-unbreakable in G, then any of its subset $A' \subseteq A$ is also (q,k)-unbreakable in G. Moreover, if A is (q,k)-unbreakable in G, then A is also (q,k)-unbreakable in any supergraph of G. For a small set A it is easy to efficiently verify whether A is (q,k)-unbreakable in G, or to find a witnessing separation.

Lemma 2.7 (♠). *Given a graph G, a set $A \subseteq V(G)$ and an integer q one can check in $\mathcal{O}(|A|^{2q+2}k(n+m))$ time whether A is (q,k)-unbreakable in G, and if not, then find a separation (X, Y) of order at most k such that $|(X \setminus Y) \cap A| > q$ and $|(Y \setminus X) \cap A| > q$.*

3 Decomposition

We now restate our decomposition theorem in a slightly stronger form that will emerge from the proof.

Theorem 3.1. *There is an $\mathcal{O}(2^{\mathcal{O}(k^2)}n^2m)$ time algorithm that, given a connected graph G together with an integer k, computes a tree decomposition (T, β) of G with at most n nodes such that the following conditions hold:*

(i) for each $t \in V(T)$, the graph $G[\gamma(t)] \setminus \sigma(t)$ is connected and $N(\gamma(t) \setminus \sigma(t)) = \sigma(t)$,

(ii) for each $t \in V(T)$, the set $\beta(t)$ is $(2^{\mathcal{O}(k)}, k)$-unbreakable in $G[\gamma(t)]$,

(iii) for each non-root $t \in V(T)$, we have that $|\sigma(t)| \leq 2^{\mathcal{O}(k)}$ and $\sigma(t)$ is $(2k,k)$-unbreakable in $G[\gamma(\text{parent}(t))]$.

3.1 Proof overview

We first give an overview of the proof of Theorem 3.1, ignoring the requirement that each adhesion is supposed to be $(2k,k)$-unbreakable. As discussed in the introduction, this property is only used to improve the running time of the algorithm, and is not essential to establish fixed-parameter tractability.

We prove the decomposition theorem using a recursive approach, similar to the standard framework used for instance by Robertson and Seymour [26] or by Marx and Grohe [16]. That is, in the recursive step we are given a graph G together with a relatively small set $S \subseteq V(G)$ (i.e., of size bounded by $2^{\mathcal{O}(k)}$), and our goal is to construct a decomposition of G satisfying the requirements of

Theorem 3.1 with an additional property that S is contained in the root bag of the decomposition. The intention is that the recursive step is invoked on some subgraph of the input graph, and the set S is the adhesion towards the decomposition of the rest of the graph.

Henceforth we focus on one recursive step, and consider three cases. In the base case, if $|S| \leq 3k$, we add an arbitrary vertex to S and repeat. In what follows, we assume $|S| > 3k$.

First, assume that S is $(2k, k)$-breakable in G, and let (X, Y) be the witnessing separation. We proceed in a standard manner (cf. [26]): we create a root bag $A := S \cup (X \cap Y)$, for each connected component C of $G \setminus A$ recurse on $(N_G[C], N_G(C))$, and glue the obtained trees as children of the root bag. It is straightforward from the definition of the witnessing separation that in every recursive call we have $|N_G(C)| \leq |S|$. Moreover, clearly $|A| \leq |S| + k$ and hence A is appropriately unbreakable.

In the last, much more interesting case the adhesion S turns out to be $(2k, k)$-unbreakable. Hence, any separation (X, Y) in G partitions S very unevenly: almost the entire set S, up to $\mathcal{O}(k)$ elements, lies on only one side of the separation. Let us call this side the "big" side, and the second side the "small" one.

The main idea now is as follows: if, for each $v \in V(G)$, we mark all important separators of size $\mathcal{O}(k)$ between v and S, then the marked vertices will separate all "small" sides of separations from the set S. Let B be the set of marked vertices and let A be the set of all vertices of G that are either in $B \cup S$, or are not separated from S by any of the considered important separator. We observe that the strong structure of important separators — in particular, the single-exponential bound on the number of important separators for one vertex v — allows us to argue that each connected component C of $G \setminus A$ that is separated by some important separator from S has only bounded number of neighbours in A. Moreover, the fact that we cut all "small" sides of separations implies that A is appropriately unbreakable in G. Hence, we may recurse, for each connected component C of $G \setminus A$ that is separated by some important separator from S, on $(N_G[C], N_G(C))$, and take A as a root bag.

The section is organised as follows. In Section 3.2 we define formally the notion of *chips*, that are parts of the graph cut out by important separators, and provide all the properties that play crucial role in Section 3.3. In Section 3.3 we also show how to proceed with the case S being unbreakable, that is, how extract the root bag containing S by cutting away all the chips. In Section 3.4 we perform some technical augmentation to ensure that the adhesions are $(2k, k)$-unbreakable. Finally in Section 3.5 we combine the obtained results and construct the main decomposition of Theorem 3.1.

3.2 Chips

In this subsection we define fragments of the graph which are easy to chip (i.e. cut out of the graph) from some given set of vertices S, and show their basic properties.

Definition 3.2 (chips). For a fixed set of vertices $S \subseteq V$, a subset $C \subseteq V$ is called a *chip*, if $G[C]$ is connected, $|N(C)| \leq 3k$, and $N(C)$ is an important $C - S$ separator.

Let $\mathcal{C}$ be the set of all inclusion-wise maximal chips.

The following lemma is straightforward from the definition of important separators.

Lemma 3.3. *For any nonempty set $C \subseteq V(G)$ such that $G[C]$ is connected, the following conditions are equivalent.*

(i) $N(C)$ is an important $C - S$ separator;

(ii) for any $v \in C$, $N(C)$ is an important $v - S$ separator;

(iii) there exists $v \in C$ such that $N(C)$ is an important $v - S$ separator.

Note also that for a connected set of vertices D and any important $D - S$ separator Z of size at most $3k$ that is disjoint with D, the set of vertices reachable from D in $G \setminus Z$ forms a chip. Lemmata 2.4 and 3.3 show how to enumerate inclusion-wise maximal chips.

Lemma 3.4 (♠). *Given a set $S \subseteq V(G)$ one can compute the set $\mathcal{C}$ of all inclusion-wise maximal chips in $\mathcal{O}(2^{\mathcal{O}(k)}n(n+m))$ time. In particular, $|\mathcal{C}| \leq 4^{3k}n$.*

Definition 3.5 (chips touching). We say that two chips $C_1, C_2 \in \mathcal{C}, C_1 \neq C_2$, *touch* each other, denoted $C_1 \sim C_2$, if $C_1 \cap C_2 \neq \emptyset$ or $E(C_1, C_2) \neq \emptyset$.

The following lemma provides an alternative definition of touching that we will find useful.

Lemma 3.6. *$C_1 \in \mathcal{C}$ touches $C_2 \in \mathcal{C}$ if and only if $N(C_1) \cap C_2 \neq \emptyset$.*

Proof. From right to left, if $v \in N(C_1) \cap C_2$ then there exists a neighbour u of v that belongs to C_1, and consequently $uv \in E(C_1, C_2)$.

From left to right, first assume $C_1 \cap C_2 \neq \emptyset$. Since $\mathcal{C}$ contains only inclusion-wise maximal chips, we have that $C_2 \setminus C_1 \neq \emptyset$. By the properties of chips, the graph $G[C_2]$ is connected, hence there is an edge between $C_2 \setminus C_1$ and $C_1 \cap C_2$ inside $G[C_2]$. This proves $N(C_1) \cap C_2 \neq \emptyset$.

In the other case, assume that $C_1 \cap C_2 = \emptyset$ but there exists $uv \in E(C_1, C_2)$ such that $u \in C_1$ and $v \in C_2$. Since $C_1 \cap C_2 = \emptyset$, it follows that $v \notin C_1$, and hence $v \in N(C_1) \cap C_2$. □

The next result provides the most important tool for bounding the size of adhesions in the constructed decomposition.

Lemma 3.7. *Any chip $C \in \mathcal{C}$ touches at most $3k \cdot 4^{3k}$ other chips of $\mathcal{C}$.*

Proof. Assume that C touches some $C' \in \mathcal{C}$. By Lemma 3.6 there exists a vertex $v \in N(C) \cap C'$. Observe that since $N(C')$ is an important $C' - S$ separator, then $N(C')$ is also an important $v - S$ separator. By Lemma 2.4 there are at most 4^{3k} important $v - S$ separators of size at most $3k$. Since $|N(C)| \leq 3k$ (by the properties of chips), we infer that C touches at most $3k \cdot 4^{3k}$ chips from $\mathcal{C}$. □

3.3 Local decomposition

Equipped with basic properties of chips we are ready to prove the main step of the decomposition part of the paper. In what follows we show that given a $(2k, k)$-unbreakable set S of size bounded in k one can find a (potentially large) unbreakable part $A \subseteq V$ of the graph, such that $S \subseteq A$ and each connected component of $G \setminus A$ is adjacent to a small number of vertices of A. In what follows, let us define

$$\eta = 3k \cdot (3k \cdot 4^{3k} + 1), \qquad \tau = (3k)^2 \cdot 8^{3k} + 2k.$$

Theorem 3.8. *There is an $\mathcal{O}(2^{O(k)}nm)$ time algorithm that, given a connected graph G together with an integer k and a $(2k, k)$-unbreakable set $S \subseteq V(G)$, computes a set $A \subseteq V(G)$ such that:*

(a) $S \subseteq A$,

(b) for each connected component D of $G \setminus A$ we have $|N_G(D)| \leq \eta$,

(c) A is (τ, k)-unbreakable in G, and

(d) if $|S| > 3k$, $G \setminus S$ is connected and $N(V(G) \setminus S) = S$, then $S \neq A$.

Proof. Let $\mathcal{C}$ be the set of inclusion-wise maximal chips, enumerated by Lemma 3.4. We define

$$A = \left(\bigcap_{C \in \mathcal{C}} V(G) \setminus N[C] \right) \cup \bigcup_{C \in \mathcal{C}} N(C).$$

In the definition we assume that when $\mathcal{C}$ is empty, then $A = V(G)$. The claimed running time of the algorithm follows directly from Lemma 3.4.

For property (a), note that no vertex of S is contained in a chip of $\mathcal{C}$, hence $S \subseteq A$. We now show property (d). Note that $N(V(G) \setminus S) = S$ and $|S| > 3k$ implies $S \neq V(G)$. Consequently, if $\mathcal{C} = \emptyset$, property (d) is straightforward. Otherwise, let $C \in \mathcal{C}$. Note that $|S| > 3k$ implies that $S \setminus N(C) \neq \emptyset$ and the connectivity of $G \setminus S$ together with $N(V(G) \setminus S) = S$ further implies that $N(C) \setminus S \neq \emptyset$. Consequently, $A \setminus S \neq \emptyset$ and property (d) is proven.

We now move to the remaining two properties.

Claim 3.9. *For any connected component D of $G \setminus A$ there exists a chip $C_1 \in \mathcal{C}$ such that $D \subseteq C_1$.*

Proof. Observe that a vertex which is not contained in any chip belongs to the set A, as it is either contained in $N(C)$ for some $C \in \mathcal{C}$ or it belongs to $V(G) \setminus N[C]$ for every $C \in \mathcal{C}$. Let D be an arbitrary connected component of $G \setminus A$ and let $v \in D$ be its arbitrary vertex. As $v \notin A$, there is a chip $C_v \in \mathcal{C}$ such that $v \in C_v$. Recall that by its definition the set A contains all the neighbours of all the chips in $\mathcal{C}$, hence $N(C_v) \cap D = \emptyset$ and by the connectivity of $G[D]$ we have $D \subseteq C_v$. ⌟

In the following claim we show that the set A satisfies property (b) of Theorem 3.8.

Lemma 3.10. *For any connected component D of $G \setminus A$ it holds that $|N(D)| \leq \eta$.*

Proof. Let D be an arbitrary connected component of $G \setminus A$. By Claim 3.9 there exists $C \in \mathcal{C}$ such that $D \subseteq C$. Intuitively each vertex of $N(D)$ belongs to the set A for one of two reasons: (i) it belongs to $N(C)$, or (ii) it is adjacent to a vertex of some other chip, which touches C. In both cases we show that there is only a bounded number of such vertices, which is formalized as follows.

Let v be any vertex of $N(D)$. Clearly $v \in N[C]$, hence we either have $v \in N(C)$ or $v \in C$. Observe that if $v \in C$, then since $v \in A$, by the definition of the set A we have $v \in N(C')$ for some $C' \in \mathcal{C}$, $C' \neq C$. Since $v \in N(C') \cap C$, then C' touches C by Lemma 3.6. We infer that $N(D) \subseteq N(C) \cup \bigcup_{C' \in \mathcal{C}, C \sim C'} N(C')$. The claimed upper bound on $|N(D)|$ follows from Lemma 3.7. □

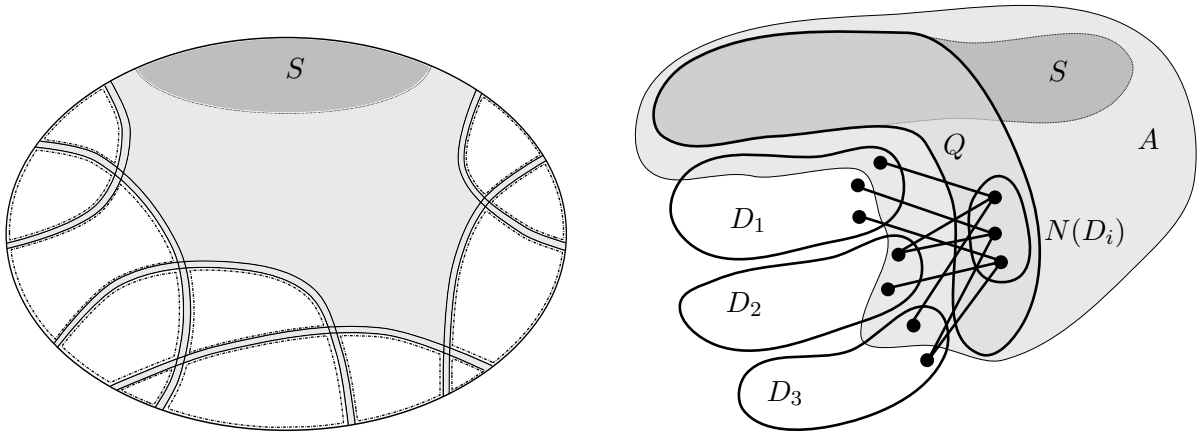
Next, we show that the set A is unbreakable. A short an informal rationale behind this property is that everything what could be easily cut out of the graph was already excluded in the definition of A.

Lemma 3.11. *The set A is (τ, k)-unbreakable.*

Proof. Assume the contrary, and let (X, Y) be a witnessing separation, i.e. we have that $|X \cap Y| \leq k$, $|(X \setminus Y) \cap A| > \tau$ and $|(Y \setminus X) \cap A| > \tau$. Since S is $(2k, k)$-unbreakable, then either $|(X \setminus Y) \cap S| \leq 2k$ or $|(Y \setminus X) \cap S| \leq 2k$. Without loss of generality we assume that $|(X \setminus Y) \cap S| \leq 2k$. Let us define a set $Q = (X \cap Y) \cup (X \cap S)$ and observe that $|Q| \leq 3k$.

Note that each connected component of $G \setminus Q$ is either entirely contained in $X \setminus Y$ or in $Y \setminus X$ (see Fig. 1a). Consider connected components of the graph $G \setminus Q$ that are contained in $X \setminus Y$ and observe that they contain at least $|((X \setminus Y) \cap A) \setminus S| > \tau - 2k$ vertices of A in total. Therefore, by grouping the connected components of $G \setminus Q$ contained in $X \setminus Y$ by their neighbourhoods in Q, we infer that there exists a set of connected components $\mathcal{D} = \{D_1, \ldots, D_r\}$, such that $\forall_{1 \leq i,j \leq r} N_G(D_i) = N_G(D_j)$ and

$$\left| \bigcup_{i=1}^{r} D_i \cap A \right| > \frac{\tau - 2k}{2^{3k}} = (3k)^2 \cdot 4^{3k}. \tag{1}$$



(a) Construction of the bag A (b) Situation in the proof of Claim 3.12.

Figure 1: Illustrations of the proof of Theorem 3.8

We now need the following claim.

Claim 3.12. *There is a subset $\mathcal{C}_0 \subseteq \mathcal{C}$, such that each $v \in \bigcup_{i=1}^{r} D_i \cap A$ belongs to some chip of $\mathcal{C}_0$, and there are at most $3k \cdot 4^{3k}$ chips in $\mathcal{C}$ that touch some chip of $\mathcal{C}_0$.*

Proof. Observe that, for each $1 \leq i \leq r$, Q is a $D_i - S$ separator (see Fig. 1a) of size at most $3k$. Therefore, for each D_i there is an important $D_i - S$ separator of size at most $3k$ disjoint with D_i, hence each D_i is contained in some chip of $\mathcal{C}$. Consider two cases.

First, assume that for each $1 \leq i \leq r$ we have $D_i \in \mathcal{C}$. As $\mathcal{C}_0$ take $\{D_1, \ldots, D_r\}$. Observe that as components D_i have the same neighbourhoods in G, then by Lemma 3.6 each chip of $\mathcal{C}$ that touches some chip D_i touches also D_1. Therefore, by Lemma 3.7 there are at most $3k \cdot 4^{3k}$ chips in $\mathcal{C}$ that touch some chip of $\mathcal{C}_0$.

In the second case assume that there exist $1 \leq i_0 \leq r$ and a chip $C \in \mathcal{C}$ such that $D_{i_0} \subsetneq C$. We shall prove that for each $1 \leq i \leq r$ we have $D_i \subseteq C$. Since C is connected and $C \setminus D_{i_0}$ is non-empty, we have that $C \cap N(D_{i_0}) \neq \emptyset$. Let $C' = C \cup \bigcup_{1 \leq i \leq r} D_i$. Clearly $C' \cap S = \emptyset$, and C' is connected since each component D_i is adjacent to every vertex of $C \cap N(D_{i_0})$. Moreover, as each D_i has the same neighbourhood in Q we have $|N(C')| \leq |N(C)| \leq 3k$ (see Fig. 1b).As $\mathcal{C}$ contains only maximal chips we have $C' = C$ and hence $\bigcup_{1 \leq i \leq r} D_i \subseteq C$. Define $\mathcal{C}_0$ as $\{C\}$. By Lemma 3.7 a single chip touches at most $3k \cdot 4^{3k}$ other chips, which finishes the proof of Claim 3.12 ⌟

Let $v \in A \cap D_i$ for some $1 \leq i \leq r$. Since v is contained in some $C' \in \mathcal{C}_0$, we have $v \notin V(G) \setminus N[C']$. Consequently, by the definition of the set A there exists a chip $C_v \in \mathcal{C}$ such that $v \in N(C_v)$. Note that $C' \neq C_v$ and $N(C_v) \cap C' \neq \emptyset$, hence by Lemma 3.6 C' touches C_v. By Claim 3.12 there are at most $3k \cdot 4^{3k}$ chips touching a chip of $\mathcal{C}_0$. As each C_v satisfies $|N(C_v)| \leq 3k$, we infer that the number of vertices of A in $\bigcup_{1 \leq i \leq r} D_i$ is at most $(3k)^2 4^{3k}$, which contradicts (1) and finishes the proof of Lemma 3.11. □

Lemma 3.10 and Lemma 3.11 ensure properties (b) and (c) of the set A, respectively. This concludes the proof of Theorem 3.8. □

3.4 Strengthening unbreakability of adhesions

So far Theorem 3.8 provides us with a construction of the bag that meets almost all the requirements, apart from $(2k, k)$-unbreakability of adhesions. For this reason, in this section we want to show

that the set A from Theorem 3.8 can be extended to a set A' in such a way that for each connected component D of $G \setminus A'$ the set $N_G(D)$ is even $(2k,k)$-unbreakable. During this extension we may weaken unbreakability of A', but if we are careful enough then this loss will be limited to a single-exponential function of k. As the results of this section are only auxiliary (they only improve the dependency on the parameter k in the subsequent dynamic programming routines from double- to single-exponential), the proofs are postponed to Section 7. In the following we let

$$\tau' = \tau + \left(\binom{\tau + k}{2} \cdot k + k \right) \cdot k\eta \,.$$

Lemma 3.13 (♠). *Let G be a graph, and $L \subseteq V(G)$ be a subset of vertices of size at least $2k+1$. Then one can in $\mathcal{O}(|L|^{4k+3}kn(n+m))$ time find a set L', $L \subseteq L'$, such that $|L' \setminus L| \leq (|L|-2k-1)\cdot k$ and for each connected component D of $G \setminus L'$, we have that $|N_G(D)| \leq |L|$ and $N_G(D)$ is $(2k,k)$-unbreakable in G.*

Theorem 3.14 (♠). *There is an $\mathcal{O}(2^{O(k^2)}nm)$ time algorithm that, given a connected graph G together with an integer k and a $(2k,k)$-unbreakable set S, computes a set $A' \subseteq V(G)$ such that:*

(a) $S \subseteq A'$,

(b) for each connected component D of $G \setminus A'$ the set $N_G(D)$ is $(2k,k)$-unbreakable, and $|N_G(D)| \leq \eta$,

(c) A' is (τ', k)-unbreakable in G,

(d) moreover, if $|S| > 3k$, $G \setminus S$ is connected and $N(V(G) \setminus S) = S$, then $S \neq A'$.

3.5 Constructing a decomposition

In this subsection we prove our main decomposition theorem, i.e. Theorem 3.1. However, for the inductive approach to work we need a bit stronger statement, where additionally we have a set $S \subseteq V(G)$ that has to be contained in the top bag of the tree decomposition. Note that Theorem 3.1 follows from the following by setting $S = \emptyset$.

Theorem 3.15. *There is an $\mathcal{O}(2^{\mathcal{O}(k^2)}n^2m)$ time algorithm that, given a connected graph G together with an integer k and a set $S \subseteq V(G)$ of size at most η such that $G \setminus S$ is connected and $N(V(G) \setminus S) = S$, computes a tree decomposition (T, β) such that S is contained in the top bag of the tree decomposition, and the following conditions are satisfied:*

(i) for each $t \in V(T)$, the graph $G[\gamma(t)] \setminus \sigma(t)$ is connected and $N(\gamma(t) \setminus \sigma(t)) = \sigma(t)$,

(ii) for each $t \in V(T)$, the set $\beta(t)$ is (τ', k)-unbreakable in $G[\gamma(t)]$,

(iii) for each non-root $t \in V(T)$, we have that $|\sigma(t)| \leq \eta$ and $\sigma(t)$ is $(2k,k)$-unbreakable in $G[\gamma(\text{parent}(t))]$.

(iv) $|V(T)| \leq |V(G) \setminus S|$.

Proof. If $|V(G)| \leq \tau'$, the algorithm creates a single bag containing the entire $V(G)$. It is straightforward to verify that such a decomposition satisfies all the required properties. Thus, in the rest of the proof we assume that $|V(G)| > \tau'$, in particular, $|V(G)| > 3k$.

Define $S' = S$ and, if $|S| \leq 3k$, add $3k + 1 - |S|$ arbitrary vertices of $V(G) \setminus S$ to S'. Note that, as $\eta > 3k$, we have $3k < |S'| \leq \eta$.

We now define a set A' as follows. First, we verify, using Lemma 2.7, whether S' is $(2k,k)$-breakable in G or not. If it turns out to be $(2k,k)$-breakable in G, we apply Lemma 3.13 to the pair (G,S'), obtaining a set which we denote by A'. Otherwise, we can use Theorem 3.14 on the pair (G,S') to obtain a set A'. Note that in both cases $S \subseteq S' \subseteq A'$ and all computations so far take $\mathcal{O}(2^{\mathcal{O}(k^2)}nm)$ time in total.

Regardless of the way the set A' was obtained, we proceed with it as follows. For each connected component D of $G \setminus A'$, we use Theorem 3.15 inductively for the graph $G[N[D]]$ and $S_D = N(D)$. Let us now verify that (a) each S_D is $(2k,k)$-unbreakable in G, (b) that the assumptions of the theorem are satisfied, and (c) that the recursive call is applied to a strictly smaller instance in the sense defined in the following.

For the first two claims, if S is $(2k,k)$-breakable, Lemma 3.13 asserts that $|S_D| \leq |S| \leq \eta$ and S_D is $(2k,k)$-unbreakable in G. Otherwise, property (b) of Theorem 3.14 ensures that $|S_D| \leq \eta$ and S_D is $(2k,k)$-unbreakable in G. The other assumptions on the set S_D in the recursive calls follow directly from the definitions of these calls.

For the last claim, we show that either $|N[D]| < |V(G)|$ or $N[D] = V(G)$ and $|D| < |V(G) \setminus S|$. Assume the contrary, that is, $D = V(G) \setminus S$ and $N(D) = S_D = S = S' = A'$. In particular, as S_D is $(2k,k)$-unbreakable in G, the set A' was obtained using Theorem 3.14. However, as $|S'| > 3k$, property (d) of Theorem 3.14 ensures that $S' \subsetneq A'$, a contradiction.

Let (T_D, β_D) be the tree decomposition obtained in the recursive call for the pair $(G[N[D]], S_D)$. Construct a tree decomposition (T,β), by creating an auxiliary node r, which will be the root of T, and attach T_D to r, by making the root r_D of T_D a child of r in T. Finally, define $\beta = \bigcup_D \beta_D$ and set $\beta(r) = A'$. A straightforward check shows that (T,β) is indeed a valid tree decomposition. We now proceed to verify its promised properties.

Clearly, $S \subseteq S' \subseteq A'$. For any connected component D of $G \setminus A'$, note that $\gamma(r_D) = N[D]$ and $\sigma(r_D) = N(D) = S_D$. This, together with inductive assumptions on recursive calls, proves properties (i) and (iii).

If A' is obtained using Lemma 3.13, then $|A'| \leq k|S'| \leq k\eta < \tau'$, hence clearly $A' = \beta(r)$ is (τ', k)-unbreakable. In the other case, property (c) of Theorem 3.14 ensures the unbreakability promised in property (ii).

It remains to bound the number of bags of (T,β); as each bag is processed in $\mathcal{O}(2^{\mathcal{O}(k^2)}nm)$ time this would also prove the promised running time bound. Note that by property (iv) for the recursive calls we have that $|V(T_D)| \leq |D|$ and, consequently, $|V(T)| \leq |V(G)\setminus A'|+1 = |V(G)\setminus S|+1-|A'\setminus S|$. To finish the proof of property (iv) it suffices to show that $S \subsetneq A'$. If $S \subsetneq S'$, the claim is straightforward. Otherwise, if $S = S'$ is $(2k,k)$-breakable, then Lemma 3.13 cannot return $A' = S'$ as $G \setminus S'$ is connected and $N(V(G) \setminus S') = S'$ is not $(2k,k)$-unbreakable. Consequently, $S' \subsetneq A'$ in this case. In the remaining case, when $S = S'$ is $(2k,k)$-unbreakable, property (d) of Theorem 3.14 ensures that $S' \subsetneq A'$. This finishes the proof of Theorem 3.15. □

4 Bisection

In this section we show a dynamic programming routine defined on the decomposition given by Theorem 3.1. When handling one bag of the decomposition, we essentially follow the approach of the high connectivity phase of "randomized contractions" [9]. That is, we apply the colour-coding technique in a quite involved fashion, to highlight the solution in a bag (relying heavily on the unbreakability of the bag), and then we analyse the outcome by a technical, but quite natural knapsack-style dynamic programming.

The section is organized as follows. First, in Section 4.1 we define an abstract problem which en-

capsulates the computational task one needs to perform in a single step of the dynamic programming procedure. This one, in turn, is presented in Section 4.2.

Through this section we mostly ignore the study of factors polynomial in the graph size in the running time of the algorithm, and we use the $\mathcal{O}^\star(\cdot)$ notation. We do not optimize the exponent of the polynomial in this dependency, as it adds unnecessary level of technicalities to the description, distracting from the main points of the reasoning, and, most importantly, is in fact less relevant to the main result of this paper — the fixed-parameter tractability of Minimum Bisection. In Section 4.3 we shortly argue how to obtain the running time promised in Theorem 1.1.

4.1 Hypergraph painting

Hypergraph Painting[a] (HP)
Input: Positive integers k, b, d, q, a multihypergraph H with hyperedges of size at most d, a partial function $\mathrm{col}_0 : V(H) \nrightarrow \{\mathbf{B}, \mathbf{W}\}$, and a function $f_F : \{\mathbf{B}, \mathbf{W}\}^F \times \{0, \ldots, b\} \to \{0, 1, \ldots, k, \infty\}$ for each $F \in E(H)$.
Goal: For each $0 \leq \mu \leq b$, compute the value w_μ,

$$w_\mu = \min_{\mathrm{col} \supseteq \mathrm{col}_0, (a_F)_{F \in E(H)}} \sum_{F \in E(H)} f_F(\mathrm{col}|_F, a_F)\,,$$

where the minimum is taken over colourings $\mathrm{col} : V(H) \to \{\mathbf{B}, \mathbf{W}\}$ extending col_0 and partitions of μ into non-negative integers $\mu = \sum_{F \in E(H)} a_F$, and the sum attains value ∞ whenever its value exceeds k.

[a] We are intentionally not using the name Hypergraph Colouring, as it has an established, and different, meaning.

We denote $n = |V(H)|$ and $m = |E(H)|$ throughout the analysis of the Hypergraph Painting problem.

We call an instance $(k, b, d, q, H, \mathrm{col}_0, (f_F)_{F \in E(H)})$ a *proper* instance of Hypergraph Painting if the following conditions hold:

- (**local unbreakability**), for each $F \in E(H)$, each $\mathrm{col} : F \to \{\mathbf{B}, \mathbf{W}\}$ marking more than $3k$ vertices of each colour, i.e. $|\mathrm{col}^{-1}(\mathbf{B})|, |\mathrm{col}^{-1}(\mathbf{W})| > 3k$, and each $0 \leq \mu \leq b$ the value $f_F(\mathrm{col}, \mu)$ equals ∞,
- (**connectivity**), for each $F \in E(H)$, each $\mathrm{col} : F \to \{\mathbf{B}, \mathbf{W}\}$ marking at least one vertex with each colour, i.e. $|\mathrm{col}^{-1}(\mathbf{B})|, |\mathrm{col}^{-1}(\mathbf{W})| > 0$, and each $0 \leq \mu \leq b$ the value $f_F(\mathrm{col}, \mu)$ is non-zero,
- (**global unbreakability**) for each $0 \leq \mu \leq b$ such that $w_\mu < \infty$ there is a witnessing colouring $\mathrm{col} : V(H) \to \{\mathbf{B}, \mathbf{W}\}$, which colours at most q vertices with one of the colours, i.e. $\min(|\mathrm{col}^{-1}(\mathbf{B})|, |\mathrm{col}^{-1}(\mathbf{W})|) \leq q$.

Note that, by local unbreakability, for proper instances each function f_F can be represented by at most $(2\sum_{i=0}^{3k} d^i) \cdot (b+1) \leq 4(b+1)d^{3k}$ values which are smaller than ∞.

We are going to use the well-established tool of fixed parameter tractability, namely the colour-coding technique of Alon, Yuster and Zwick [1]. A standard method of derandomizing the colour-coding technique is to use *splitters* of Naor et al. [24]. We present our algorithm already in its derandomized form, and for this reason we use the following abstraction of splitters.

Lemma 4.1 (Lemma I.1 of [9]). *Given a set U of size n, and integers $0 \leq a, b \leq n$, one can in $O(2^{O(\min(a,b)\log(a+b))} n \log n)$ time construct a family $\mathcal{F}$ of at most $O(2^{O(\min(a,b)\log(a+b))} \log n)$ subsets of U, such that the following holds: for any sets $A, B \subseteq U$, $A \cap B = \emptyset$, $|A| \leq a$, $|B| \leq b$, there exists a set $S \in \mathcal{F}$ with $A \subseteq S$ and $B \cap S = \emptyset$.*

In our dynamic programming routine we will use the following operators, to make the description of the actual algorithm more concise.

Definition 4.2. For two functions $g, h : \{0, \ldots, b\} \to \{0, 1, \ldots, k, \infty\}$ we define functions $g \oplus h, \min(g, h)$ as follows:

$$(g \oplus h)(\mu) = \min_{\mu_1 + \mu_2 = \mu} g(\mu_1) + h(\mu_2)\,,$$
$$\min(g, h)(\mu) = \min(g(\mu), h(\mu)),$$

where each integer larger than k is treated as ∞.

Note that given two functions g, h one can compute $g \oplus h$ in $\mathcal{O}(b^2)$ time, and $\min(g, h)$ in $\mathcal{O}(b)$ time.

Lemma 4.3. *There is an $\mathcal{O}^\star(q^{\mathcal{O}(k)} \cdot d^{O(k^2)})$ time algorithm solving the* Hypergraph Painting *problem for proper instances.*

Proof. First, let us fix the value of μ, $0 \leq \mu \leq b$. Our goal is to compute a single value w_μ[1]. It is enough to compute the correct value of w_μ assuming

$$w_\mu < \infty\,. \tag{2}$$

Consequently, by the global unbreakability property let $\mathrm{col}_{opt} : V(H) \to \{\mathbf{B}, \mathbf{W}\}$ be a colouring witnessing the value w_μ, that colours at most q vertices with one of the colours. Without loss of generality let us assume that

$$|\mathrm{col}_{opt}^{-1}(\mathbf{W})| \leq q\,, \tag{3}$$

as the other case $|\mathrm{col}^{-1}(\mathbf{B})| \leq q$ is symmetric.

Observe that by the local unbreakability property for each $F \in E(H)$ there are at most $\ell = 4d^{3k} = d^{O(k)}$ possible colourings leading to a value of f_F which is different than ∞. For each $F \in E(H)$ let us order the possible bichromatic colourings of F arbitrarily, and for $1 \leq i \leq \ell$ let $\mathrm{col}_{F,i}$ be the i-th of the possible colouring which is bichromatic on F (if the number of such colourings is smaller than ℓ we append the sequence with arbitrary bichromatic colourings).

We want to assign each $F \in E(H)$ to be in one of the following states:

- F is definitely monochromatic,
- F is either monochromatic, or should be coloured as in $\mathrm{col}_{F,i}$ for a fixed $1 \leq i \leq \ell$.

Formally, for an assignment $p : E(H) \to \{0, \ldots, \ell\}$ by $p(F) = 0$ we express the "definitely monochromatic" state, and by $p(F) = i > 0$ we express the "either monochromatic or i-th type of bichromatic colouring" state.

Let $E_{white} = \{F \in E(H) : \mathrm{col}_{opt}(F) = \{\mathbf{W}\}\}$ be the multiset of monochromatic edges of $E(H)$ coloured all white with respect to col_{opt}. Moreover let $E_0 \subseteq E_{white}$ be any spanning forest of the hypergraph $(V(H), E_{white})$. By (3) we have $|E_0| \leq q$. Let $E_1 \subseteq E(H)$ be the set of edges which are bichromatic with respect to col_{opt}. Note that by the the connectivity property together with (2) we have $|E_1| \leq k$.

We call an assignment p *good*, with respect to col_{opt}, if:

[1] Actually our algorithm after a minor modification computes all the values w_μ at once, however for the sake of simplicity we focus on a single value of μ, at the cost of higher polynomial factor.

- for each $F \in E_1$ we have $p(F) > 0$ and $\mathrm{col}_{opt}|_F = \mathrm{col}_{F,p(F)}$,
- for each $F \in E_0$ we have $p(F) = 0$.

Let $\mathcal{F}$ be a family constructed by the algorithm of Lemma 4.1 for the universe $E(H)$ and integers k, q in $\mathcal{O}^\star(q^{O(k)})$ time. By the properties of $\mathcal{F}$ there exists $S_0 \in \mathcal{F}$ such that $E_1 \subseteq S_0$ and $E_0 \cap S_0 = \emptyset$. We iterate through all possible $S \in \mathcal{F}$; in one of the cases we have $S = S_0$.

In the second level of derandomization we use the standard notion of perfect families. An (N, r)-*perfect family* is a family of functions from $\{1, 2, \ldots, N\}$ to $\{1, 2, \ldots, r\}$, such that for any subset $X \subseteq \{1, 2, \ldots, N\}$ of size r, one of the functions in the family is injective on X. Naor et al. [24] gave an explicit construction of an (N, r)-perfect family of size $\mathcal{O}(e^r r^{\mathcal{O}(\log r)} \log N)$ using $\mathcal{O}(e^r r^{\mathcal{O}(\log r)} N \log N)$ time. We construct a $(|S|, k)$-perfect family $\mathcal{D}$ of size $\mathcal{O}(2^{\mathcal{O}(k)} \log |S|)$. Assuming that we consider the case when $S = S_0$, there exists a function $\delta_0 \in \mathcal{D}$, $\delta_0 : S_0 \to \{1, \ldots, k\}$, such that δ_0 is injective on E_1. We iterate through all possible functions $\delta \in \mathcal{D}$; providing that $S = S_0$, in one case we have $\delta = \delta_0$.

Finally, we guess, by trying all $\ell^{\mathcal{O}(k)} = d^{\mathcal{O}(k^2)}$ possibilities, a function $\delta' : \{1, \ldots, k\} \to \{1, \ldots, \ell\}$. In the case where $S = S_0$ and $\delta = \delta_0$, for at least one such δ' we have that $\delta'(\delta_0(F)) = i$ holds for each $F \in E_1$, where $\mathrm{col}_{opt}|_F = \mathrm{col}_{F,i}$. Summing up, in one of the $\mathcal{O}^\star(q^{\mathcal{O}(k)} \cdot d^{\mathcal{O}(k^2)})$ cases we will end up having a good assignment $p : E(H) \to \{0, \ldots, \ell\}$ at hand.

For an assignment $p : E(H) \to \{0, \ldots, \ell\}$ define an auxiliary undirected simple graph L_p, with a vertex set $V(L) = V(H)$. For each edge $F \in E(H)$ such that $p(F) = 0$ make F a clique in L_p. For each edge $F \in E(H)$ such that $p(F) = i > 0$ make the sets $\mathrm{col}_{F,i}^{-1}(\mathbf{B})$, $\mathrm{col}_{F,i}^{-1}(\mathbf{W})$ cliques in L_p.

Claim 4.4. *If p is a good assignment, then all the vertices contained in the same connected component of L_p are coloured with the same colour in* col_{opt}.

Proof. Follows directly from the assumption that p is a good assignment and from the definition of the graph L_p. □

Claim 4.5. *Let p be a good assignment. If D is a connected component of L_p coloured white by* col_{opt}*, then each edge $F \in E(H)$ such that $F \cap D \neq \emptyset$ and $F \setminus D \neq \emptyset$, belongs to E_1.*

Proof. Assume that $F \notin E_1$, that is, F is monochromatic in col_{opt}. As $F \cap D \neq \emptyset$, col_{opt} needs to colour all elements of F white, and $D \in E_{white}$. However, since E_0 is a spanning forest of the hypergraph $(V(H), E_{white})$ and p is a good assignment, then all the elements of F are contained in the same connected component of L_p, and consequently $F \subseteq D$. □

From now on let us assume that p is a good assignment.

Let us modify the assignment p as follows; note that we modify also the graph L_p along with p. As long as possible perform one of the following two operations, preferring the first one over the second one:

1. If there exists an edge $F \in E(H)$ such that $F \subseteq D$ for some connected component of L_p and $p(F) > 0$, then set $p(F) = 0$.
2. If there exist vertices $v_1, v_2 \in D$ (potentially $v_1 = v_2$), and hyperedges $F_1, F_2 \in E(H)$ (potentially $F_1 = F_2$) intersecting a connected component D of L_p, such that $F_1 \setminus D \neq \emptyset$, $F_2 \setminus D \neq \emptyset$, $p(F_1) = i > 0$, $p(F_2) = j > 0$, and $\mathrm{col}_{F_1,i}(v_1) = \mathbf{W}$ and $\mathrm{col}_{F_2,j}(v_2) = \mathbf{B}$, then set $p(F_1) = 0$.

As in each round the number of edges of $E(H)$ assigned zeros is strictly increasing, the process finishes in polynomial time.

Claim 4.6. *After each step p remains a good assignment.*

Proof. First assume that the first type of operation was performed. By Claim 4.4 all the vertices of D are coloured with the same colour by col_{opt}, hence in particular all the vertices of F are coloured with the same colour in col_{opt} and definitely $F \notin E_1$. Hence it is safe to set $p(F) = 0$.

Now assume that the second type of operation was performed. We want to show that F_1 is monochromatic in col_{opt}. Assume the contrary, i.e., $F_1 \in E_1$. Since p was a good assignment (before the operation) we have $\mathrm{col}_{opt}|_{F_1} = \mathrm{col}_{F_1,p(F_1)}$, which together with Claim 4.4 implies that all the vertices of D are coloured white by col_{opt}. However by Claim 4.5 this means that $F_2 \in E_1$, and as p is a good assignment this means that col_{opt} colours all the vertices of D black, a contradiction. □

We call a connected component D of L_p a *black component* if there exists an edge $F_2 \in E(H)$, such that $p(F_2) = i > 0$ and $\mathrm{col}^{-1}_{F_2,i}(\mathbf{B}) \cap D \neq \emptyset$. Otherwise we call D a *potentially white* component. The next two claims show that these names are in fact meaningful.

Claim 4.7. *For any black component D of L_p, col_{opt} colours all vertices of D black.*

Proof. Let F be an edge witnessing D is a black component. Note that $F \not\subseteq D$, as otherwise the first operation would be applicable to F. By Claim 4.4, col_{opt} colours all vertices of D in the same colour. If this colour is white, then, by Claim 4.5, $F \in E_1$, and, as p is a good assignment, $\mathrm{col}_{F,p(F)} = \mathrm{col}_{opt}|_F$. However, this contradicts the assumption that $\mathrm{col}_{F,p(F)}$ colours some vertex of D black, and, consequently, col_{opt} colours D black. □

Claim 4.8. *Let D be a potentially white component of L_p and let $E_D \subseteq E(H)$ be the subset of edges with non-empty intersection with D. Exactly one of the following conditions holds:*

- *col_{opt} colours all vertices of D white, and for each edge $F \in E_D$ either*
  - *$F \not\subseteq D$, $p(F) > 0$, $F \in E_1$, $\mathrm{col}_{opt}|_F = \mathrm{col}_{F,p(F)}$, or*
  - *$F \subseteq D$, $p(F) = 0$ and col_{opt} colours all vertices of F white;*
- *col_{opt} colours D and each each edge of E_D entirely black.*

Proof. Let $E' \subseteq E_D$ be the subset of edges of E_D which are not fully contained in D. By Claim 4.4, col_{opt} colours D monochromatically.

Assume first that col_{opt} colours all vertices of D white, and consider $F \in E_D$. If $F \subseteq D$ then $p(F) = 0$ by the application of the first operation, and $F \in E_{white}$. If $F \not\subseteq D$ then, by Claim 4.5, $F \in E_1$. Since p is good, $p(F) > 0$ and $\mathrm{col}_{opt}|_F = \mathrm{col}_{F,p(F)}$.

We are left with the case when col_{opt} colours all vertices of D black. Consider $F \in E_D$. If $p(F) = 0$ then the assumption that p is good implies that col_{opt} colours F monochromatically; as $F \cap D \neq \emptyset$ then F is coloured black by col_{opt}. If $p(F) = i > 0$ then, since D is potentially white, $\mathrm{col}_{F,i}(v) = \mathbf{W} \neq \mathrm{col}_{opt}(v)$ for any $v \in F \cap D$. Consequently, $\mathrm{col}_{opt}|_F \neq \mathrm{col}_{F,i}$ and, since p is good, $F \notin E_1$. Therefore col_{opt} colours F monochromatically, and, since $F \cap D \neq \emptyset$, it colours F black. □

Let E_{black} be the set of all edges of $E(H)$ contained in black components of L_p. The following claim states that we may consider sets E_{black} and E_D for different potentially white components D independently.

Claim 4.9. *Every edge $F \in E(H)$ belongs to exactly one of the sets: to E_{black} or to one of the sets E_D for potentially white connected components D of L_p.*

Proof. Assume first that $F \subseteq D$ for some connected component D of L_p. Then either D is black and $F \in E_{black}$, or D is potentially white and $F \in E_D$.

Assume now that F is not entirely contained in any connected component of L_p. By the construction of L_p, we have that $p(F) = i > 0$ and F intersects exactly two different components D_1, D_2 of L_p, such that w.l.o.g. $\mathrm{col}_{F,i}^{-1}(\mathbf{W}) = D_1 \cap F$ and $\mathrm{col}_{F,i}^{-1}(\mathbf{B}) = D_2 \cap F$. To prove the claim it suffices to show that (a) D_1 is potentially white, and (b) D_2 is black, as then F will belong only to E_{D_1} among the sets present in the statement of the claim. For (a), observe that otherwise the second operation would set $p(F) = 0$, and (b) follows directly from the definition of being black. □

Armed with Claims 4.7, 4.8 and 4.9, we proceed to presenting the algorithm. First, we need to include the constraints imposed by the colouring col_0. To this end, for any hyperedge F, $0 \leq \mu \leq b$ and any colouring $\mathrm{col} : F \to \{\mathbf{B}, \mathbf{W}\}$ we set

$$\hat{f}_F(\mathrm{col}, \mu) = \begin{cases} \infty & \text{if } \exists_{v \in F}\ \mathrm{col}(v) \neq \mathrm{col}_0(v) \\ f_F(\mathrm{col}, \mu) & \text{otherwise.} \end{cases}$$

That is, we set the cost of colouring F with col as ∞ whenever col conflicts with col_0 on some vertex.

Now we handle edges contained entirely in black components. For an edge $F \in E_{black}$ let $\hat{f}_F^{black} : \{0, \ldots, b\} \to \{0, 1, \ldots, k, \infty\}$ be the function

$$\hat{f}_F^{black}(\mu) = \hat{f}_F(\{\mathbf{B}\}^F, \mu)\,.$$

Let $t : \{0, \ldots, b\} \to \{0, 1, \ldots, k, \infty\}$ be a function such that $t(0) = 0$ and $t(\mu) = \infty$ for $\mu > 0$. For each edge $F \in E_{black}$ we update the function t by setting $t := t \oplus \hat{f}_F^{black}$. It remains to process all the edges $E(H) \setminus E_{black}$.

Consider all the white components D of L_p one by one. Let $t_1, t_2 : \{0, \ldots, b\} \to \{0, 1, \ldots, k, \infty\}$ be functions such that $t_1(0) = t_2(0) = 0$ and $t_1(\mu) = t_2(\mu) = \infty$ for $\mu > 0$. We want to make t_1 represent the case when all the edges of E_D are black, while t_2 represent the other case of Claim 4.8. First, for each edge $F \in E_D$ set $t_1 := t_1 \oplus \hat{f}_F(\{\mathbf{B}\}^F, \cdot)$. Moreover, for each edge $F \in E_D$ such that $p(F) = 0$ do $t_2 := t_2 \oplus \hat{f}_F(\{\mathbf{W}\}^F, \cdot)$, while for each edge $F \in E_D$ such that $p(F) > 0$ do $t_2 := t_2 \oplus \hat{f}_F(\mathrm{col}_{F,p(F)}, \cdot)$.

Finally make the update

$$t := \min(t \oplus t_1, t \oplus t_2)\,.$$

At the end of the process the value $t(\mu)$ equals w_μ and the correctness of our algorithm follows from Claim 4.7, Claim 4.8, and Claim 4.9. □

4.2 Dynamic programming

In this section we show that by constructing a tree decomposition from Theorem 3.1 and invoking the algorithm of Lemma 4.3 one can solve the Minimum Bisection problem in $\mathcal{O}^\star(2^{O(k^3)})$ time, proving Theorem 1.1 .

Proof of Theorem 1.1. First note that, without loss of generality, we may focus on the following variant: the input graph G is required to be connected, and our goal is to partition $V(G)$ into parts A and B of prescribed size minimizing $|E(A, B)|$. The algorithm for the classic Minimum Bisection problem follows from a standard knapsack-type dynamic programming on connected components of the input graph.

As the input graph is connected, we may use Theorem 3.1. Let (T, β) be a tree decomposition constructed by the algorithm of Theorem 3.1 in $\mathcal{O}^\star(2^{O(k^2)})$ time.

As usual for tree decompositions, we will use a dynamic programming approach. For a node $t \in V(T)$ of the tree decomposition, an integer μ, $0 \leq \mu \leq n$, and a colouring $\mathrm{col}_0 : \sigma(t) \to \{\mathbf{B}, \mathbf{W}\}$ satisfying

$$\min(|\mathrm{col}_0^{-1}(\mathbf{B})|, |\mathrm{col}_0^{-1}(\mathbf{W})|) \leq 3k\,, \tag{4}$$

we consider a variable $x_{t,\mathrm{col}_0,\mu}$.

The variable $x_{t,\mathrm{col}_0,\mu}$ equals the minimum cardinality of a set $Z \subseteq E(G[\gamma(t)])$, such that there exists a colouring $\mathrm{col} : \gamma(t) \to \{\mathbf{B}, \mathbf{W}\}$, where $\mathrm{col}|_{\sigma(t)} = \mathrm{col}_0$, no edge of $E(G[\gamma(t)]) \setminus Z$ is incident to two vertices of different colours in col, and the total number of white vertices equals μ, i.e. $|\mathrm{col}^{-1}(\mathbf{W})| = \mu$. Additionally if it is impossible to find such a colouring col, or the number of edges one needs to include in Z is greater than k, then we define $x_{t,\mathrm{col}_0,\mu} = \infty$. The restriction (4) of col_0 will be used to optimize the running time.

As Theorem 3.1 upper bounds the cardinality of $\sigma(t)$ by $2^{O(k)}$, the total number of values $x_{t,\mathrm{col}_0,\mu}$ we want to compute is $\mathcal{O}(2^{O(k^2)}n^2)$. Note that having all those values is enough to solve the considered variant of the Minimum Bisection problem as the minimum possible size of the cut $E(A,B)$ equals $x_{r,\emptyset,a}$, where r is the root of (T,β), $\emptyset$ plays the role of the single colouring of $\sigma(r) = \emptyset$ and a is the prescribed size of one part of the partition we are looking for. The value $x_{r,\emptyset,a}$ attains ∞ if any feasible cut $E(A,B)$ is of size larger than k. We will compute the values $x_{t,\cdot,\cdot}$ in a bottom-up manner, that is our computation is performed for a node $t \in V(T)$ only after all the values $x_{t',\cdot,\cdot}$ for $t' \prec t$ have been already computed.

Consider a fixed $t \in V(T)$ and a colouring $\mathrm{col}_0 : \sigma(t) \to \{\mathbf{B}, \mathbf{W}\}$ satisfying (4). In what follows we show how to find all the values $x_{t,\mathrm{col}_0,\cdot}$ by solving a single proper instance of the Hypergraph Painting problem. Create an auxiliary hypergraph H, with a vertex set $V(H) = \beta(t)$ and the following set of edges; in the following we use Iverson notation, i.e., $[\varphi]$ is equal to 1 if the condition φ is true and 0 otherwise.

(a) For each vertex $v \in \beta(t)$ add to H a hyperedge $F = \{v\}$, and define a function $f_F : \{\mathbf{B}, \mathbf{W}\}^F \times \{0, \ldots, n\} \to \{0, 1, \ldots, k, \infty\}$

$$f_F(\mathrm{col}_F, \mu) = \begin{cases} 0 & \text{if } \mu = [\mathrm{col}_F(v) = \mathbf{W}], \\ \infty & \text{otherwise.} \end{cases}$$

We introduce those edges in order to keep track of the number of white vertices in $\beta(t)$.

(b) For each edge $uv \in E(G[\beta(t)])$ add to H a hyperedge $F = \{u, v\}$, and define a function $f_F : \{\mathbf{B}, \mathbf{W}\}^F \times \{0, \ldots, n\} \to \{0, 1, \ldots, k, \infty\}$

$$f_F(\mathrm{col}_F, \mu) = \begin{cases} [\mathrm{col}_F(u) \neq \mathrm{col}_F(v)] & \text{if } \mu = 0, \\ \infty & \text{otherwise.} \end{cases}$$

We introduce those edges in order to keep track of the number of edges with endpoints of different colours in $G[\beta(t)]$.

(c) For each $t' \in V(T)$ which is a child of t in the tree decomposition add to H a hyperedge $F = \sigma(t')$, and define a function $f_F : \{\mathbf{B}, \mathbf{W}\}^F \times \{0, \ldots, n\} \to \{0, 1, \ldots, k, \infty\}$

$$f_F(\mathrm{col}_F, \mu) = \begin{cases} \infty & \text{if } \min(|\mathrm{col}_F^{-1}(\mathbf{B})|, |\mathrm{col}_F^{-1}(\mathbf{W}))| > 3k \text{ or } x_{t',\mathrm{col}_F,\mu+\mu_0} = \infty \\ x_{t',\mathrm{col}_F,\mu+\mu_0} - x_0 & \text{otherwise,} \end{cases}$$

where $\mu_0 = |\mathrm{col}_F^{-1}(\mathbf{W})|$, and $x_0 = |\{uv \in E(G[\sigma(t')]) : \mathrm{col}_F(u) \neq \mathrm{col}_F(v)\}|$. Less formally, we are shifting values x_0 and μ_0 in order not to overcount white vertices of $\sigma(t')$ and edges of $G[\sigma(t')]$ having endpoints of different colours in col_F, as a vertex of $\sigma(t')$ might appear in several bags, and similarly an edge of $G[\sigma(t')]$ may have both endpoints in several bags being children of t'.

Note that each of the edges of H is of size at most η (by Theorem 3.1), hence

$$I = (k, n, \eta, q, H, \mathrm{col}_0, (f_F)_{F \in E(H)})$$

is an instance of the Hypergraph Painting problem for any q, which we are about to define.

Claim 4.10. *Let $(w_\mu)_{0 \leq \mu \leq n}$ be the solution for the instance I of the* Hypergraph Painting *problem. Then for any $0 \leq \mu \leq n$ we have $x_{t,\mathrm{col}_0,\mu} = w_\mu$.*

Moreover for any colouring $\mathrm{col} : \beta(t) \to \{\mathbf{B}, \mathbf{W}\}$ witnessing $w_\mu \leq k$ there is an extension $\mathrm{col}' : \gamma(t) \to \{\mathbf{B}, \mathbf{W}\}$, such that $\mathrm{col}'|_{\beta(t)} = \mathrm{col}$, and the number of bichromatic edges of $G[\gamma(t)]$ with respect to col' equals w_μ.

Proof. Fix an arbitrary $0 \leq \mu \leq n$. First, we show that $x_{t,\mathrm{col}_0,\mu} \geq w_\mu$. Note that the inequality trivially holds for $x_{t,\mathrm{col}_0,\mu} = \infty$, hence let us assume $x_{t,\mathrm{col}_0,\mu} \leq k$ and let $\mathrm{col} : \gamma(t) \to \{\mathbf{B}, \mathbf{W}\}$ be a colouring such that

- $\mathrm{col}|_{\sigma(t)} = \mathrm{col}_0$,
- $|Z| \leq k$, where $Z = \{uv \in E(G[\gamma(t)]) : \mathrm{col}(u) \neq \mathrm{col}(v)\}$,
- $|\mathrm{col}^{-1}(\mathbf{W})| = \mu$.

Recall that Theorem 3.1 ensures that for any child t' of t in the tree decomposition the adhesion $\sigma(t')$ is $(2k, k)$-unbreakable in $G[\gamma(t)]$. Therefore,

$$\min(|\mathrm{col}^{-1}(\mathbf{B}) \cap \sigma(t')|, |\mathrm{col}^{-1}(\mathbf{W}) \cap \sigma(t')|) \leq 3k\,,$$

as otherwise $(X = N_{G[\gamma(t)]}[\mathrm{col}^{-1}(\mathbf{B})], Y = \mathrm{col}^{-1}(\mathbf{W}))$ would be a separation of $G[\gamma(t)]$ of order at most k with $|(X \setminus Y) \cap \sigma(t')|, |(Y \setminus X) \cap \sigma(t')| > 2k$, contradicting the fact that $\sigma(t')$ is $(2k, k)$-unbreakable in $G[\gamma(t)]$. Consequently, the values $x_{t',\mathrm{col}|_{\sigma(t')},\cdot}$ are well-defined, i.e., $\mathrm{col}|_{\sigma(t')}$ satisfies (4). Furthermore, observe that

$$x_{t',\mathrm{col}|_{\sigma(t')},|\mathrm{col}^{-1}(\mathbf{W}) \cap \gamma(t')|} \leq |Z \cap E(G[\gamma(t')])|, \tag{5}$$

which is witnessed by the colouring $\mathrm{col}|_{\gamma(t')}$. For $F \in E(H)$ define $a_F = |\mathrm{col}^{-1}(\mathbf{W}) \cap (\gamma(t') \setminus \sigma(t'))|$. Next, we verify that $\mathrm{col}|_{\beta(t)}$ and $(a_F)_{F \in E(H)}$ certify that $x_{t,\mathrm{col}_0,\mu} \geq w_\mu$. We split the contributions of edges of $E(H)$ to the sum $\sum_{F \in E(H)} f_F(\mathrm{col}|_F, a_F)$ into three summands, according to the types of edges of $E(H)$. The edges of type (a) do not contribute to the sum at all. The edges of type (b) contribute exactly $|Z \cap E(G[\beta(t)])|$, while the edges of type (c) contribute exactly $\sum_{t'} x_{t',\mathrm{col}|_{\sigma(t')},a_F+|\mathrm{col}^{-1}(\mathbf{W}) \cap \sigma(t')|} - |Z \cap E(G[\sigma(t')])| \leq \sum_{t'} |Z \cap (E(G[\gamma(t')] \setminus E(G[\sigma(t')])))|$, where the sum is over all children t' of t and the inequality follows from (5). This means that each edge of Z is counted exactly once, so the total contribution is at most $|Z| = x_{t,\mathrm{col}_0,\mu}$.

In the other direction, we want to show $x_{t,\mathrm{col}_0,\mu} \leq w_\mu$. As in the previous case, for $w_\mu = \infty$ the inequality trivially holds. Hence, we assume $w_\mu \leq k$. Let $\mathrm{col} : \beta(t) \to \{\mathbf{B}, \mathbf{W}\}$ be a colouring and $\sum_{F \in E(H)} a_F$ be a partition of μ witnessing the value of w_μ, i.e., satisfying

- $\mathrm{col}|_{\sigma(t)} = \mathrm{col}_0$,

- $\sum_{F \in E(H)} a_F = \mu$,
- $w_\mu = \sum_{F \in E(H)} f_F(\mathrm{col}|_F, a_F)$.

Our goal is to extend the colouring col on $\gamma(t) \setminus \beta(t)$, so that the total number of white vertices equals μ and the number of bichromatic edges equals w_μ. Initially set $\mathrm{col}' = \mathrm{col}$ and consider children t' of t in the tree decomposition one by one. Let $F = \sigma(t') \in E(H)$ be the type (c) edge of H. Since $w_\mu \leq k$, we have $f_F(\mathrm{col}|_F, \mu_F) \leq k$, and by the definition of f_F

$$f_F(\mathrm{col}|_F, \mu_F) = x_{t', \mathrm{col}|_F, \mu_F - \mu_0} - x_0 \,,$$

where $\mu_0 = |\mathrm{col}^{-1}(\mathbf{W}) \cap \sigma(t')|$, and $x_0 = |\{uv \in E(G[\sigma(t')]) : \mathrm{col}(u) \neq \mathrm{col}(v)\}|$. Let $\mathrm{col}_F : \gamma(t') \to \{\mathbf{B}, \mathbf{W}\}$ be the colouring witnessing the value $x_{t', \mathrm{col}|_F, \mu_F - \mu_0}$. Note that col_F is consistent with col on $F = \sigma(t')$, so we can update col' by setting $\mathrm{col}' = \mathrm{col}' \cup \mathrm{col}_F$.

Observe that the edges of $E(H)$ of type (a) together with shifting by μ_0 ensure that col' colours exactly μ vertices white. Finally, the edges of $E(H)$ of type (b) together with shifting by x_0 ensure that col' has exactly w_μ bichromatic edges, which shows $x_{t, \mathrm{col}_0, \mu} \leq w_\mu$. As $\mathrm{col}'|_{\beta(t)} = \mathrm{col}$ the last part of the claim follows as well. □

The previous claim shows that solving the Hypergraph Painting instance I is enough to find the values $x_{t, \mathrm{col}_0, \cdot}$, however in the previous section we have only shown how to solve proper instances of Hypergraph Painting. Therefore, we show that there is a small enough value of q, such that I becomes a proper instance.

Claim 4.11. *There is $q = 2^{O(k)}$ such that I is a proper instance of the* Hypergraph Painting *problem.*

Proof. For the hyperedges $F \in E(H)$ of size at most two the local unbreakability property is trivially satisfied, while for all the other hyperedges $F = \sigma(t')$ local unbreakability follows directly from the definition of f_F.

By Theorem 3.1 each $G[\gamma(t')] \setminus \sigma(t')$ is connected and $N(\gamma(t') \setminus \sigma(t')) = \sigma(t')$, which means that the graph $G[\gamma(t')] \setminus E(G[\sigma(t')])$ is connected, and consequently $x_{t', \mathrm{col}_F, \cdot} > 0$ for any colouring col_F which uses both colours (as we need to remove at least one edge). This proves the connectivity property.

Recall, that by Theorem 3.1 the set $\beta(t)$ is (τ', k)-unbreakable in $G[\gamma(t)]$ for some $\tau' = 2^{O(k)}$. Let $q = \tau' + k$ and let $(w_\mu)_{0 \leq \mu \leq n}$ be a solution for the instance I of the Hypergraph Painting problem. Consider an arbitrary $0 \leq \mu \leq n$ such that $w_\mu \leq k$. We want to show, that there exists a witnessing colouring $\mathrm{col} : \beta(t) \to \{\mathbf{B}, \mathbf{W}\}$ certifying the global unbreakability. In fact we will show that any colouring $\mathrm{col} : \beta(t) \to \{\mathbf{B}, \mathbf{W}\}$ witnessing w_μ satisfies

$$\min(|\mathrm{col}^{-1}(\mathbf{B})|, |\mathrm{col}^{-1}(\mathbf{W})|) \leq q = \tau' + k \,.$$

By Claim 4.10[2] there is an extension col' of col, having $w_\mu \leq k$ bichromatic edges of $G[\gamma(t)]$. Note that $(X = N_{G[\gamma(t)]}[\mathrm{col}'^{-1}(\mathbf{B})], Y = \mathrm{col}'^{-1}(\mathbf{B}))$ is a separation of $G[\gamma(t)]$ of order at most k, hence by (τ', k)-unbreakability of $\beta(t)$ we have

$$\min(|(X \setminus Y) \cap \beta(t)|, |(Y \setminus X) \cap \beta(t)|) \leq \tau' \,.$$

However $|\mathrm{col}^{-1}(\mathbf{B})| \leq |(X \setminus Y) \cap \beta(t)| + k$ and $|\mathrm{col}^{-1}(\mathbf{W})| \leq |(Y \setminus X) \cap \beta(t)| + k$, which implies

$$\min(|\mathrm{col}^{-1}(\mathbf{B})|, |\mathrm{col}^{-1}(\mathbf{W})|) \leq k + \tau' = q \,,$$

proving the global unbreakability property. □

[2]Note that to use Claim 4.10 we do not require that I is proper.

By Claim 4.11 we can use Lemma 4.3 and in $\mathcal{O}^\star(q^{\mathcal{O}(k)} \cdot \eta^{\mathcal{O}(k^2)}) = \mathcal{O}^\star(2^{\mathcal{O}(k^3)})$ time compute the values w_μ for each $0 \leq \mu \leq n$. At the same time Claim 4.10 shows that $x_{t,\mathrm{col}_0,\mu} = w_\mu$ for each $0 \leq \mu \leq n$. Since the number of nodes of $V(T)$ is at most $|V(G)|$ and the number of colourings obeying (4) is $2^{\mathcal{O}(k^2)}$ the whole dynamic programming routine takes $\mathcal{O}^\star(2^{\mathcal{O}(k^3)})$ time. Consequently Theorem 1.1 follows. □

4.3 Dependency on the size of G in the running time

Here we argue about the factors polynomial in the size of G in the running time of the algorithm.

We first note that we may assume that $m = |E(G)| = \mathcal{O}(kn)$, by applying the sparsification technique of Nagamochi and Ibaraki [23].

Lemma 4.12 ([23]). *Given an undirected graph G and an integer k, in $O(k(|V(G)| + |E(G)|))$ time we can obtain a set of edges $E_0 \subseteq E(G)$ of size at most $(k+1)(|V(G)|-1)$, such that for any edge $uv \in E(G) \setminus E_0$ in the graph $(V(G), E_0)$ there are at least $k+1$ edge-disjoint paths between u and v.*

Proof. The algorithm performs exactly $k+1$ iterations. In each iteration it finds a spanning forest F of the graph G, adds all the edges of F to E_0 and removes all the edges of F from the graph G.

Observe that for any edge uv remaining in the graph G, the vertices u and v are in the same connected components in each of the forests found. Hence in each of those forests we can find a path between u and v; thus, we obtain $k+1$ edge-disjoint paths between u and v. □

The above lemma allows us to sparsify the graph, so that it contains $\mathcal{O}(kn)$ edges, and any edge cut of size at most k remains in the graph, while any edge cut with at least $k+1$ edges after sparsification still has at least $k+1$ edges. Therefore applying Lemma 4.12 gives us an equivalent instance $(V(G), E_0)$ and consequently the construction of the decomposition takes $\mathcal{O}(2^{\mathcal{O}(k^2)}n^3)$ time.

There are at most n bags of the decomposition, which adds a $\mathcal{O}(n)$ factor to the running time. In each bag t, we consider $\mathcal{O}(\eta^{\mathcal{O}(k)})$ colourings of the adhesion $\sigma(t)$; hence, there are $\mathcal{O}(\eta^{\mathcal{O}(k)}n)$ calls to the procedure solving Hypergraph Painting.

In each call, we have $V(H) = \beta(t)$ and $|E(H)| = \mathcal{O}(n+m) = \mathcal{O}(kn)$, as we have a hyperedge for each vertex and edge of $\beta(t)$ as well as an edge for each child of t in the decomposition. As discussed in Section 4.1, each function f can be represented by giving $\mathcal{O}(\eta^{\mathcal{O}(k)}n)$ values different than ∞.

Note that we do not need to perform the entire algorithm for Hypergraph Painting for each value of μ independently. Instead, we may perform it only once, and return w_μ to be the minimum $t(\mu)$ among all branches of the algorithm.

By Lemma 4.1 and the construction of perfect families of [24], there are $\mathcal{O}(2^{\mathcal{O}(k^3)}\log^2 n)$ choices of the assignment p, and they can be enumerated in $\mathcal{O}(2^{\mathcal{O}(k^3)}n\log^2 n)$ time.

For each assignment p, we need to perform the two operations exhaustively. To speed them up, instead of maintaining the entire graph L_p, we keep only its connected components: each vertex of H knows its connected component, and the connected component knows its size and its vertices. In this manner, by enumerating the smaller component, we can merge two connected components in amortized $\mathcal{O}(\log n)$ time, as each vertex changes the connected components it belongs to $\mathcal{O}(\log n)$ times. Consequently we may initiate the graph L_p in $\mathcal{O}(\eta kn + n\log n)$ time, as we have to iterate over $\mathcal{O}(kn)$ edges of size η each, and the total time needed to merge connected components is $\mathcal{O}(n\log n)$.

To apply the operations, we maintain the following auxiliary information. Each hyperedge F stores a set of the connected components of L_p it intersects (note that this set is of size at most 2). Once this set changes its cardinality from 2 to 1, the first operation starts to be applicable on F. As each vertex changes its connected component $\mathcal{O}(\log n)$ times, each list is updated at most $\mathcal{O}(\log n)$ times, which gives $\mathcal{O}(kn\log n)$ time in total.

For the second operation, we need to maintain, for each connected component D of L_p, a list $T(D, \mathbf{B})$ of hyperedges F such that $F \cap D \neq \emptyset$, $F \setminus D \neq \emptyset$, $p(F) > 0$ and $\mathrm{col}_{F,p(F)}(v) = \mathbf{B}$ for some $v \in F \cap D$; analogously we define a list $T(D, \mathbf{W})$. Once both lists are non-empty, the second operation is applicable. As each hyperedge is of size at most η, all lists can be recomputed in $\mathcal{O}(\eta k n)$ time, whenever the set of the connected components of the graph L_p changes: each hyperedge F inserts itself into at most η lists.

We infer that the operations can be exhaustively applied in $\mathcal{O}(2^{\mathcal{O}(k)}n^2)$ time for a fixed assignment p. Also, including the constraints imposed by the colouring col_0, i.e., obtaining the functions $\hat{f}_F(\mathrm{col}, \mu)$, can be done in $\mathcal{O}(2^{\mathcal{O}(k^2)}n^2)$ time.

We now move to the analysis of the final knapsack-type dynamic programming routine. We first show that the $\oplus$ operation can be performed using $\mathcal{O}(k^2)$ applications of the Fast Fourier Transform, taking total time $\mathcal{O}(k^2 b \log b)$, instead of the naive $\mathcal{O}(b^2)$ time bound. Consider two functions $t_1, t_2 \in \{0, \ldots, b\} \to \{0, \ldots, k, \infty\}$. For $i \in \{1, 2\}$ and $0 \leq j \leq k$ by $p_{i,j}$ we define the polynomial

$$p_{i,j}(x) = \sum_{0 \leq \mu \leq b} [t_i(\mu) = j] x^\mu \,.$$

Note that if $(t_1 \oplus t_2)(\mu) \neq \infty$, then $(t_1 \oplus t_2)(\mu)$ is equal to the smallest j, such that for some partition $j = j_1 + j_2$ the coefficient in front of the monomial x^μ in the polynomial $p_{1,j_1} \cdot p_{2,j_2}$ is non-zero. Therefore we can compute $t_1 \oplus t_2$ in $\mathcal{O}(k^2 b \log b)$ time. There are $\mathcal{O}(1)$ such operations per each edge of $E(H)$. Consequently, the final dynamic programming algorithm takes $\mathcal{O}(2^{O(k)}n^2 \log n)$ time.

We conclude that the total running time is $\mathcal{O}(2^{\mathcal{O}(k^3)}n^3 \log^3 n)$, as promised in Theorem 1.1.

5 Weighted variant

In this section we sketch how using our approach one can solve the following weighted variant of the Minimum Bisection problem:

Theorem 5.1. *Given a graph G with edge weights $w : E(G) \to \mathbb{R}$ and an integer k, one can in $\mathcal{O}^\star(2^{k^3})$ time find a partition of $V(G)$ into sets A and B minimizing $\sum_{e \in E(A,B)} w(e)$ subject to $||A| - |B|| \leq 1$ and $|E(A, B)| \leq k$, or state that such a partition does not exist.*

Proof. Essentially, we follow the same approach as in the previous section, except that in all dynamic programming tables we need to add an additional dimension to control the size of the constructed cut $E(A, B)$, and store the weight of the cut as the value of the entry in the DP table.

In some more details, for a fixed bag t, a colouring $\mathrm{col}_0 : \sigma(t) \to \{\mathbf{B}, \mathbf{W}\}$ satisfying (4), and integers $0 \leq \mu \leq n$ and $0 \leq \xi \leq k$ we consider a variable $x_{t,\mathrm{col}_0,\mu,\xi} \in \mathbb{R} \cup \{+\infty\}$ that equals the minimum possible value of $\sum_{e \in E(\mathrm{col}^{-1}(\mathbf{B}), \mathrm{col}^{-1}(\mathbf{W}))} w(e)$ among colourings $\mathrm{col} : \gamma(t) \to \{\mathbf{B}, \mathbf{W}\}$ satisfying:

- $\mathrm{col}|_{\sigma(t)} = \mathrm{col}_0$,
- $|\mathrm{col}^{-1}(\mathbf{W})| = \mu$, and
- $|E(\mathrm{col}^{-1}(\mathbf{B}), \mathrm{col}^{-1}(\mathbf{W}))| = \xi$.

The value $+\infty$ is attained if no such colouring exists.

Analogously, we modify the Hypergraph Painting problem to match the aforementioned definition of the values $x_{t,\mathrm{col}_0,\mu,\xi}$. That is, it takes as an input functions $f_F : \{\mathbf{B}, \mathbf{W}\}^F \times \{0, \ldots, b\} \times \{0, \ldots, k\} \to \mathbb{R} \cup \{+\infty\}$, where we require value $+\infty$ for any colouring that violates the local unbreakability constraint. For each $0 \leq \mu \leq b$ and $0 \leq \xi \leq k$ we seek for a value $w_{\mu,\xi} \in \mathbb{R} \cup \{+\infty\}$

defined as a minimum, among all colourings col extending col_0, and all possible sequences $(a_F)_{F\in E(H)}$ and $(b_F)_{F\in E(H)}$ such that $\sum_F a_F = \mu$ and $\sum_F b_F = \xi$, of

$$\sum_{F\in E(H)} f_F(\mathrm{col}|_F, a_F, b_F).$$

The knapsack-type dynamic programming of Section 4.1 is adjusted in a natural way, and the remaining reasoning of Section 4.1 remains unaffected by the weights. Consequently, the adjusted Hypergraph Painting problem can be solved in $\mathcal{O}^\star(q^{\mathcal{O}(k)}\cdot d^{O(k^2)})$ time.

It is straightforward to check that the adjusted Hypergraph Painting problem corresponds again to the task of handling one bag in the tree decomposition of the input graph. To finish the proof note that the value we are looking for equals $\min_{0\leq\xi\leq k} x_{r,\emptyset,\lfloor n/2\rfloor,\xi}$, where r is the root of the tree decomposition. □

6 α-edge-separators

In this section we argue that the algorithm of Section 4 can be extended to show the following:

Theorem 6.1. *Given an n-vertex graph G, a real $\alpha\in(0,1)$ and an integer k, one can in $\mathcal{O}(2^{\mathcal{O}(k^3)}n^{\mathcal{O}(1/\alpha)})$ time decide if there exists a set X of at most k edges of G such that each connected component of $G\setminus X$ has at most αn vertices.*

To prove Theorem 6.1, we need the following lemma that can be seen as a generalization of Lemma 7.3 of [2].

Lemma 6.2. *Let $\alpha\in(0,1)$ be a real constant and let $a_1,a_2,\ldots,a_n\in[0,\alpha]$ be reals such that $\sum_{\ell=1}^n a_\ell = 1$. Then one can partition numbers $a_1,a_2,\ldots,a_n$ into $2\lceil\frac{1}{\alpha}\rceil-1$ groups (possibly empty), such that the sum of numbers in each group is at most α.*

Proof. Let $q=\lceil\frac{1}{\alpha}\rceil$. For $\ell=0,1,\ldots,n$, let $b_\ell=\sum_{i=1}^{\ell}a_i$. For $j=1,2,\ldots,q-1$, let i_j be the unique index such that $b_{i_j-1}\leq j\cdot\alpha$ and $b_{i_j}>j\cdot\alpha$. Let us denote also $i_0=0$ and $i_q=n+1$; then also $b_{i_0}\geq 0\cdot\alpha$ and $b_{i_q-1}\leq q\cdot\alpha$. Define the following groups:

$$\begin{Bmatrix} \{a_1,a_2,\ldots,a_{i_1-1}\},\\ \{a_{i_1}\},\\ \{a_{i_1+1},a_{i_1+2},\ldots,a_{i_2-1}\},\\ \{a_{i_2}\},\\ \ldots\\ \{a_{i_{q-2}+1},a_{i_{q-2}+2},\ldots,a_{i_{q-1}-1}\},\\ \{a_{i_{q-1}}\},\\ \{a_{i_{q-1}+1},a_{i_{q-1}+2},\ldots,a_n\} \end{Bmatrix}.$$

For every group of form $\{a_{i_j+1},a_{i_j+2},\ldots,a_{i_{j+1}-1}\}$ we have that

$$\sum_{\ell=i_j+1}^{i_{j+1}-1} a_\ell = b_{i_{j+1}-1}-b_{i_j}\leq (j+1)\alpha - j\alpha = \alpha.$$

On the other hand, for every group of form $\{a_{i_j}\}$ we have that $a_{i_j}\leq\alpha$ by the assumption that $a_{i_j}\in[0,\alpha]$. Hence, the formed groups satisfy the required properties. □

The following corollary is immediately implied by Lemma 6.2.

Corollary 6.3. *Let $\alpha \in (0,1)$ be a real constant and let H be a graph on n vertices. Then the following conditions are equivalent:*

(a) Each connected component of H has at most αn vertices.

(b) There exists a partition of $V(H)$ into ζ possibly empty sets $A_1, A_2, \ldots, A_\zeta$, where $\zeta = 2\left\lceil \frac{1}{\alpha} \right\rceil - 1$, such that $|A_i| \leq \alpha n$ for each $i = 1, 2, \ldots, \zeta$ and no edge of H connects two vertices from different parts.

Equipped with Corollary 6.3, we may now describe how to modify the algorithm of Section 4 to prove Theorem 6.1. Most of the modifications are straightforward, hence we just sketch the consecutive steps.

Proof of Theorem 6.1. By Corollary 6.3, we may equivalently seek for a colouring of $V(G)$ into $\zeta = 2\left\lceil \frac{1}{\alpha} \right\rceil - 1$ colours, such that at most k edges connect vertices of different colours. Essentially, we now proceed as in Section 4, but, instead of colouring vertices into black and white, we use ζ colours, and we keep track of the number of vertices coloured in each colour.

In some more details, for a fixed bag t, a colouring $\mathrm{col}_0 : \sigma(t) \to \{1, \ldots, q\}$ satisfying

$$\exists_{1 \leq c \leq \zeta} \ |\mathrm{col}_0^{-1}(c)| \geq |\sigma(t)| - 3k \tag{6}$$

and a function $\mu : \{1, \ldots, \zeta\} \to \{0, \ldots, n\}$ we consider a variable $x_{t,\mathrm{col}_0,\mu} \in \{0, 1, \ldots, k, \infty\}$ that equals the minimum possible number of edges with endpoints coloured by different colours by a colouring col, among all colourings $\mathrm{col} : \gamma(t) \to \{1, 2, \ldots, \zeta\}$ satisfying:

- $\mathrm{col}|_{\sigma(t)} = \mathrm{col}_0$, and
- for each $1 \leq c \leq \zeta$, $|\mathrm{col}^{-1}(c)| = \mu(c)$.

The value ∞ is attained if any such colouring yields more than k edges with endpoints of different colours.

Recall that, for any bag t, the adhesion $\sigma(t)$ is $(2k, k)$-unbreakable. Similarly as in Claim 4.10, we infer that if in a colouring $\mathrm{col} : \gamma(t) \to \{1, \ldots, \zeta\}$ at most k edges have endpoints painted in different colours, it needs to colour all but at most $3k$ vertices of $\sigma(t)$ with a single colour. This motivates condition (6). Note that this requirement is only needed to obtain $2^{\mathrm{poly}(k)}$ dependency on k, and, if it is omitted, the dependency will become doubly-exponential.

We now modify the Hypergraph Painting problem to match the aforementioned definition of the values $x_{t,\mathrm{col}_0,\mu}$. That is, the problem takes as an input functions $f_F : \{1, \ldots, \zeta\}^F \times \{0, \ldots, b\}^\zeta \to \{0, 1, \ldots, k, \infty\}$. For each $\mu : \{1, \ldots, \zeta\} \to \{0, \ldots, b\}$ we seek for a value $w_\mu \in \{0, 1, \ldots, k, \infty\}$ defined as a minimum, among all colourings $\mathrm{col} : V(H) \to \{1, \ldots, \zeta\}$ extending col_0, and all possible sequences $(a_F^c)_{F \in E(H), 1 \leq c \leq \zeta}$ such that $\sum_F a_F^c = \mu(c)$ for each $1 \leq c \leq \zeta$, of

$$\sum_{F \in E(H)} f_F(\mathrm{col}|_F, (a_F^c)_{1 \leq c \leq \zeta}).$$

The value of ∞ is attained whenever the sum exceeds k.

In the local unbreakability constraint we require that a value different than ∞ can be attained only if all but at most $3k$ elements of F are coloured in a single colour. This corresponds to the previously discussed condition (6) on valid colourings col_0 of an adhesion $\sigma(t)$. The connectivity

requirement states that $f_F(\mathrm{col}, \alpha)$ is non-zero whenever col uses at least two colours: the corresponding colouring of the subgraph $\gamma(t)$ (as in the proof of Claim 4.10) needs to colour the endpoints of at least one edge with different colours, as $\gamma(t)$ is connected. The global unbreakability constraint requires that whenever $w_\mu < \infty$, there is a witnessing colouring col that colours all but at most $\tau' + k$ vertices with a single colour. This follows from the fact that, in our decomposition, $\beta(t)$ is (τ', k)-unbreakable, so any colouring of $\gamma(t)$ that colours endpoints of at most k edges with different colours needs to paint all but at most $\tau' + k$ vertices of $\beta(t)$ with the same colour.

The core spirit of the reasoning of Section 4.1 remains in fact unaffected by this change. However, for sake of clarity, we now describe the changes in more details. We apply colour-coding to paint the hyperedges with assignment $p : E(H) \to \{0, \ldots, \ell\}$, where $p(F) = 0$ means "definitely monochromatic" and $p(F) = i > 0$ means "monochromatic or coloured according to the colouring $\mathrm{col}_{F,i} : F \to \{1, 2, \ldots, \zeta\}$". The colourings $\mathrm{col}_{F,i}$ are required to comply with the (new) unbreakability constraint, thus there are $\eta^{\mathcal{O}(k)}\zeta^{\mathcal{O}(k)}$ such colourings. We guess a colour — call it *black* — that will be the dominant colour in $\beta(t)$. We require that for all hyperedges that are not monochromatic in the solution col_{opt} we have $p(F) = i > 0$ and $\mathrm{col}_{F,i} = \mathrm{col}_{opt}|_F$ (i.e., we have guessed the correct colouring of F), and the "definitely monochromatic" hyperedges span a hyperforest of the graph $(V(H), E_{not\ black})$, where $E_{not\ black}$ consists of all not-black monochromatic hyperedges in the colouring col_{opt}.

For a hyperedge F, we insert to L_p all edges of $F \times F$ if $p(F) = 0$ and all edges between the vertices of the same colour of $\mathrm{col}_{F,i}$ if $p(F) = i > 0$. It is straightforward to verify that, if p is guessed correctly, then the following holds:

1. all connected components of L_p are painted monochromatically in col_{opt} (cf. Claim 4.4);
2. if a connected component D of L_p is not painted black in col_{opt}, then all hyperedges F such that $F \cap D \neq \emptyset$ and $F \setminus D \neq \emptyset$ are not coloured monochromatically by col_{opt} and, consequently, their colourings $\mathrm{col}_{F,p(F)}$ conforms with col_{opt} (cf. Claim 4.5).

We may now classify any connected component D of L_p as "definitely black" (there exists a hyperedge F with $p(F) > 0$ such that $\mathrm{col}_{F,p(F)}$ colours at least one vertex of D black) and "potentially not black" (otherwise). By the aforementioned discussion and a reasoning analogous to Claim 4.7, a definitely black component is painted black by col_{opt}.

The clean-up operations on p are defined as follows:

1. for any hyperedge F completely contained in some component D of L_p, F is monochromatic in col_{opt}, so set $p(F) = 0$;
2. for any hyperedge F with $p(F) > 0$, if $\mathrm{col}_{F,p(F)}$ colours some vertex of a definitely black connected component of L_p with a colour different than black, set $p(F) = 0$ as the guess on $\mathrm{col}_{F,p(F)}$ is clearly incorrect.

After the operations are performed exhaustively, it is straightforward to verify that an analogue of Claim 4.8 holds, stating that for any potentially not black component D, either col_{opt} colours D with not-black colour, being consistent with all hyperedges F with $p(F) > 0$ that intersect D, or col_{opt} colours D and all intersecting hyperedges black.

However, it is no longer true that the components of D may be considered independently in the final knapsack-type dynamic programming. Indeed, if there exists a hyperedge F that intersects two potentially not black components D_1 and D_2 (and, hence, $p(F) > 0$), then col_{opt} paints D_1 black if and only if it paints D_2 black as well. Consequently, we need to adjust the knapsack-type DP in the following way. A black component is painted black, so we can proceed with them as previously. Two potentially not black components D_1 and D_2 are *entangled* if there exists a hyperedge F intersecting

both of them. Now, observe that all components of each connected component of the entanglement relation make a joint decision on whether they are painted black or not, and these decisions are independent between each other: the decision in one connected component of the entanglement relation does not influence the decision in another one. This allows us to adjust he knapsack-type dynamic programming of Section 4.1, considering in a single step all hyperedges intersecting all connected components of L_p contained in a single connected component of the entanglement relation.

It is straightforward to check that the adjusted Hypergraph Painting problem corresponds again to the task of handling one bag in the tree decomposition of the input graph. To finish the proof note that the minimum size of the cut we are looking for equals

$$\min_{\mu:\{1,\ldots,q\}\to\{0,\ldots,\lfloor \alpha n\rfloor\}} x_{r,\emptyset,\mu},$$

where r is the root of the tree decomposition, as long as the cut has size at most k. □

7 Postponed proofs from Sections 2 and 3

Proof of Lemma 2.7. Our algorithm guesses, by trying all possibilities, two disjoint subsets $X_0, Y_0 \subseteq A$ of $q+1$ vertices each. Having fixed X_0 and Y_0 we may, in $\mathcal{O}(k(n+m))$ time by applying $(k+1)$ rounds of Ford-Fulkerson algorithm, find a minimum $X_0 - Y_0$ separator in G, or conclude that its size is larger than k. If a separator Z of size at most k exists, then obtain a separation (X', Y') as follows: set $X' \cap Y' = Z$, add connected components of $G \setminus Z$ intersecting X_0 to $X' \setminus Y'$, add connected components intersecting Y_0 to $Y' \setminus X'$, and distribute all the other connected component arbitrarily between $X' \setminus Y'$ and $Y' \setminus X'$. Observe that since $|X' \cap Y'| \leq k$, $|X' \setminus Y'| \geq |X_0| = q+1$, and $|Y' \setminus X'| \geq |Y_0| = q+1$, then separation (X', Y') witnesses that A is (q,k)-breakable, and thus can be output by the algorithm. If for none of the pairs (X_0, Y_0) admits a $X_0 - Y_0$ separator of size at most k, then we conclude that A is (q,k)-unbreakable.

It remains to argue that if A is (q,k)-breakable, then for some pair (X_0, Y_0) the minimum $X_0 - Y_0$ separator has size at most k. Indeed, let (X, Y) be any separation of order at most k witnessing that A is (q,k)-breakable, and let $X_0 \subseteq X \setminus Y$ and $Y_0 \subseteq Y \setminus X$ be any subsets of size $q+1$. Then $X \cap Y$ is an $X_0 - Y_0$ separator of size at most k. □

Proof of Lemma 3.4. For any $v \in V$, we use Lemma 2.4 to enumerate the set $\mathcal{Z}_v$ of all important $v - S$ separators of size at most $3k$. Recall that for any $Z \in \mathcal{Z}_v$, the set $R_{G\setminus Z}(v)$ is the vertex set of the connected component of $G \setminus Z$ containing v. Define $\mathcal{A}_v = \{R_{G\setminus Z}(v) : Z \in \mathcal{Z}_v\}$ and let $\mathcal{C}_v$ be the set of inclusion-wise maximal elements of $\mathcal{A}_v$. By Lemma 3.3 we infer that if some chip $C \in \mathcal{A}_v$ is not inclusion-wise maximal, then there exists $C' \in \mathcal{A}_v$ such that $C \subsetneq C'$. Therefore, we have that $\mathcal{C} = \bigcup_{v \in V(G)} \mathcal{C}_v$.

As $|\mathcal{Z}_v| \leq 4^{3k}$ for any $v \in V(G)$, the bound on $|\mathcal{C}|$ follows. For each $v \in V(G)$, the sets $\mathcal{Z}_v$, $\mathcal{A}_v$ and $\mathcal{C}_v$ can be computed in $\mathcal{O}(2^{\mathcal{O}(k)}(n+m))$ time in a straightforward manner. The computation of $\mathcal{C} = \bigcup_{v \in V(G)} \mathcal{C}_v$ in $\mathcal{O}(2^{\mathcal{O}(k)} n(n+m))$ time can be done by inserting all the elements of $\bigcup_{v \in V(G)} \mathcal{C}_v$ into a prefix tree (trie), each in $\mathcal{O}(n)$ time, and ignoring encountered duplicates. □

Proof of Lemma 3.13. We prove the lemma by induction on $|L|$, with the following two base cases. If $L = V(G)$, clearly we may return $L' = L$. In the second base case we assume that L is $(2k,k)$-unbreakable in G, which can be checked in $\mathcal{O}(|L|^{4k+2}k(n+m))$ time using Lemma 2.7. Then for each connected component D of $G \setminus L$ we have that $N_G(D) \subseteq L$, and thus $|N_G(D)| \leq |L|$ and $N_G(D)$ is also $(2k,k)$-unbreakable in G. Hence we can set $L' = L$, and since $|L| \geq 2k+1$, we have that $|L' \setminus L| \leq (|L| - 2k - 1) \cdot k$.

Now let us assume that L is $(2k,k)$-breakable in G, and hence there exists a separation (X,Y) of G such that $|X\cap Y|\leq k$, $|(X\setminus Y)\cap L|>2k$ and $|(Y\setminus X)\cap L|>2k$, found by the algorithm of Lemma 2.7. We use inductively Lemma 3.13 for the pair $(G_1=G[X], L_1=(X\cap L)\cup(X\cap Y))$ and for the pair $(G_2=G[Y], L_2=(Y\cap L)\cup(X\cap Y))$, to obtain sets L_1' and L_2', respectively. Note here that $|L_1|,|L_2|\geq 2k+1$ and $|L_1|,|L_2|<|L|$. Define $L'=L_1'\cup L_2'$. Each connected component D of $G\setminus L'$ is either a connected component of $G_1\setminus L_1'$ and is adjacent only to L_1', or is a connected component of $G_2\setminus L_2'$ and is adjacent only to L_2'. Assume without of loss of generality the first case. By inductive assumption we infer that $|N_{G_1}(D)|\leq |L_1|$ and $N_{G_1}(D)$ is $(2k,k)$-unbreakable in G_1, and since $N_{G_1}(D)=N_G(D)$, $|L_1|<|L|$, and G_1 is a subgraph of G, then it follows that $|N_G(D)|\leq |L|$ and $N_G(D)$ is $(2k,k)$-unbreakable in G. It remains to argue that the cardinality of $L'\setminus L$ is not too large. Observe that

$$L'\setminus L\subseteq (L_1'\setminus L_1)\cup(L_2'\setminus L_2)\cup(X\cap Y)\,;$$

therefore, by induction we have

$$\begin{aligned}|L'\setminus L| &\leq (|L_1|-2k-1)\cdot k+(|L_2|-2k-1)\cdot k+k\\ &\leq (|L_1|+|L_2|-4k-1)\cdot k\\ &\leq (|L|+2|X\cap Y|-4k-1)\cdot k\\ &\leq (|L|-2k-1)\cdot k\,.\end{aligned}$$

Let us now bound the running time of the recursion. Clearly, as the size of the set L decreases in the recursive calls, the depth of the recursion is at most $|L|$. Moreover, note that any vertex may appear in $V(G)\setminus L$ in at most one recursive call (G,L) at any fixed level of the recursion tree. Hence, there are at most $|L|n$ recursive calls that do not correspond to the first base case, and, consequently, at most $2|L|n+1$ recursive calls in total. As each recursive call takes $\mathcal{O}(|L|^{4k+2}k(n+m))$ time, the promised running time bound follows. □

Proof of Theorem 3.14. We start by finding the set A by running the algorithm Theorem 3.8. Next, for each connected component D of $G\setminus A$ using Lemma 2.7 we check whether $N(D)$ is $(2k,k)$-breakable in G. By Theorem 3.8, the cardinality of $N(D)$ is bounded by η, hence all tests take total time $\mathcal{O}(\eta^{4k+2}knm)=\mathcal{O}(2^{\mathcal{O}(k^2)}nm)$ time. Note that if $N(D)$ is $(2k,k)$-breakable in G, then in particular $|N(D)|>2k$, hence we can use Lemma 3.13 for the pair $(G[N[D]], L_D=N(D))$; let L_D' be the obtained set. As $|L_D|\leq\eta$, the algorithm of Lemma 3.13 runs in $\mathcal{O}(\eta^{4k+3}k|N[D]|m)$ time for a fixed component D, and total time taken by calls to Lemma 3.13 is:

$$\sum_D \mathcal{O}(\eta^{4k+3}k(|D|+|N(D)|)m)\leq \mathcal{O}(\eta^{4k+3}km)\cdot\left(\sum_D|D|+\sum_D\eta\right)=\mathcal{O}(\eta^{4k+4}knm)=\mathcal{O}(2^{\mathcal{O}(k^2)}nm).$$

In the case when $N(D)$ is $(2k,k)$-unbreakable, let $L_D=L_D'=N(D)$. Define $A'=A\cup(\bigcup_D L_D')$, where the union is taken over all the connected components D of $G\setminus A$.

Since $S\subseteq A\subseteq A'$, we have that $S\subseteq A'$, and, moreover, the property (d) follows directly from property (d) of Theorem 3.8. Moreover, as $|L_D|\leq\eta$ for each connected component D of $G\setminus A$, then by Lemma 3.13 for each connected component D' of $G\setminus A'$ we also have $|N_G(D')|\leq\eta$. The fact that $N_G(D')$ is $(2k,k)$-unbreakable in G follows directly from Lemma 3.13. It remains to show that A' is (τ',k)-unbreakable in G.

Consider any separation (X,Y) of G of order at most k. By Theorem 3.8 the set A is (τ,k)-unbreakable, hence either $|(X\setminus Y)\cap A|\leq\tau$ or $|(Y\setminus X)\cap A|\leq\tau$, and without loss of generality assume the former. As (X,Y) is an arbitrary separation of order at most k, to show that A' is (τ',k)-unbreakable it suffices to prove that $|(X\setminus Y)\cap(A'\setminus A)|\leq\binom{\tau+k}{2}\cdot k+k)\cdot k\eta$.

Note that $A' \setminus A \subseteq \bigcup_D L'_D \setminus L_D$. As for each D we have $|L'_D \setminus L_D| \leq k\eta$ by Lemma 3.13, to finish the proof of Theorem 3.14 we are going to show that there are at most $\binom{\tau+k}{2} \cdot k + k$ connected components D of $G \setminus A$ such that $D \cap (X \setminus Y) \neq \emptyset$ and $L'_D \neq L_D$. As (X, Y) is of order at most k, there are at most k connected components D of $G \setminus A$ intersecting $X \cap Y$. Hence we restrict our attention to connected components D of $G \setminus A$, such that $D \subseteq X \setminus Y$, which in turn implies $N(D) \subseteq A \cap X$. Recall that if $L'_D \neq L_D$ for such a connected component D, then $N(D)$ is $(2k, k)$-breakable in G, and hence there exist two vertices $v_a, v_b \in N(D) \subseteq A \cap X$, such that the minimum vertex cut separating v_a and v_b in G is at most k. However, such a pair of vertices v_a, v_b may be simultaneously contained in neighbourhoods of at most k connected components D, since each component D adjacent both to v_a and to v_b contributes with at least one path between them. As $|A \cap X| \leq \tau + k$, the theorem follows. □

8 Conclusions

In this paper we have settled the parameterized complexity of Minimum Bisection. Our algorithm also works in the more general setting when the edges are weighted, when the vertex set is to be partitioned into a constant number of parts rather than only two, and when the cardinality of each of the parts is given as input.

The core component of our algorithm is a new decomposition theorem for general graphs. Intuitively, we show that it is possible to partition any graph in a tree-like manner using small separators so that each of the resulting pieces cannot be broken any further. This uncovered structure is very natural in the context of cut-problems, and we strongly believe that our decomposition theorem will find many further algorithmic applications.

Having settled the parameterized complexity of Minimum Bisection it is natural to ask whether the problem also admits a *polynomial kernel*, i.e. a polynomial-time preprocessing algorithm that would reduce the size of the input graph to some polynomial of the budget k. This question, however, has been already resolved by van Bevern et al. [27], who showed that Minimum Bisection does not admit a polynomial kernel unless coNP $\subseteq$ NP/poly. We conclude with a few intriguing open questions.

(a) Can the running time of our algorithm be improved? In particular, does there exist an algorithm for Minimum Bisection with running time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$, that is with linear dependence on the parameter in the exponent?

(b) The running time dependence of our algorithm on the input size is roughly cubic. Is it possible to obtain a fixed-parameter tractable algorithm with quadratic, or even nearly-linear running time dependence on input size? Note that the best known algorithm for graphs of bounded treewidth has quadratic dependence on the input size [17].

(c) Are the parameters in the decomposition theorem tight? For example, is it possible to lower the adhesion size from $2^{\mathcal{O}(k)}$ to polynomial in k? Similarly, can one make the bags $(k^{\mathcal{O}(1)}, k)$-unbreakable rather than $(2^{\mathcal{O}(k)}, k)$-unbreakable? Is it possible to achieve both simultaneously? We remark that if the latter question has a positive answer, this would improve the parameter dependence in the running time of our algorithm for Minimum Bisection to $k^{\mathcal{O}(k)}$.

(d) Is it possible to compute our decomposition faster, say in $2^{\mathcal{O}(k \log k)}n^{\mathcal{O}(1)}$ or even in $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time? Currently the main bottleneck is the very simple Lemma 2.7, which we are unable to speed up.

Acknowledgements

We would like to thank Rajesh Chitnis, Fedor Fomin, MohammadTaghi Hajiaghayi and M. S. Ramanujan for earlier discussions on this subject.

References

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[2] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. A $\mathcal{O}(c^k n)$ 5-approximation algorithm for treewidth. *CoRR*, abs/1304.6321, 2013. Extended abstract to appear in the proceedings of FOCS 2013.

[3] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7(2):171–191, 1987.

[4] T. N. Bui, C. Heigham, C. Jones, and F. T. Leighton. Improving the performance of the Kernighan-Lin and simulated annealing graph bisection algorithms. In *DAC*, pages 775–778, 1989.

[5] T. N. Bui and A. Peck. Partitioning planar graphs. *SIAM J. Comput.*, 21(2):203–215, 1992.

[6] T. N. Bui and L. C. Strite. An ant system algorithm for graph bisection. In *GECCO*, pages 43–51, 2002.

[7] J. Carmesin, R. Diestel, F. Hundertmark, and M. Stein. Connectivity and tree structure in finite graphs. *CoRR*, abs/1105.1611, 2011.

[8] J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.

[9] R. H. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *FOCS*, pages 460–469, 2012.

[10] R. Diestel. *Graph Theory*. Springer, 2005.

[11] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM J. Comput.*, 31(4):1090–1118, 2002.

[12] U. Feige, R. Krauthgamer, and K. Nissim. Approximating the minimum bisection size (extended abstract). In *STOC*, pages 530–536, 2000.

[13] U. Feige and M. Mahdian. Finding small balanced separators. In *STOC*, pages 375–384, 2006.

[14] M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. Freeman New York, 1979.

[15] M. Grohe, K. Kawarabayashi, D. Marx, and P. Wollan. Finding topological subgraphs is fixed-parameter tractable. In *STOC*, pages 479–488, 2011.

[16] M. Grohe and D. Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. In *STOC*, pages 173–192, 2012.

[17] K. Jansen, M. Karpinski, A. Lingas, and E. Seidel. Polynomial time approximation schemes for max-bisection on planar and geometric graphs. *SIAM J. Comput.*, 35(1):110–119, 2005.

[18] K. Kawarabayashi and M. Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *FOCS*, pages 160–169, 2011.

[19] S. Khot and N. K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into l_1. In *FOCS*, pages 53–62, 2005.

[20] D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.

[21] D. Marx, B. O'Sullivan, and I. Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30, 2013.

[22] D. Marx and I. Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. In L. Fortnow and S. P. Vadhan, editors, *STOC*, pages 469–478. ACM, 2011.

[23] H. Nagamochi and T. Ibaraki. A Linear-Time Algorithm for Finding a Sparse k-Connected Spanning Subgraph of a k-Connected Graph. *Algorithmica*, 7(5&6):583–596, 1992.

[24] M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *Proc. of FOCS'95*, pages 182–191, 1995.

[25] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *STOC*, pages 255–264, 2008.

[26] N. Robertson and P. D. Seymour. Graph minors XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

[27] R. van Bevern, A. E. Feldmann, M. Sorge, and O. Suchý. On the parameterized complexity of computing graph bisections. In A. Brandstädt, K. Jansen, and R. Reischuk, editors, *WG*, volume 8165 of *Lecture Notes in Computer Science*, pages 76–87. Springer, 2013.